

OK
OK
OK
.NLIST

```

30|( CONSTANTS AND I/O PORT DEFINITIONS )
31|( RAM ORIGIN AND SPECIAL VGS VERBS )
32|( VGS random, ranger )
33|( VGS random, RND, RANGER, RANGERND )
34|( UPDATE COLOR MAP -- QUICKLY )
35|( VGS FLOOD )
36|( VGS FILL )
37|( VGS write routines pattern representation )
38|( VGS pattern board and magic equates )
39|( RELABS ) F= relabs SUBR ffreelabs (ASSEMBLE
40|( VGS RELABS )
41|( VGS write )
42|( VGS write con't. )
43|( VGS write con't. )
44|( VGS writep )
45|( VGS WRITEP )
46|( VGS WRITE )
47|( FRAME, UNFRAME MACROS )
48|( SPECIAL RELABS FOR BOX )
49|( PATTERN BOARD BOX COMMAND )
50|( X Y XS YS MODE BOX ) HEX
51|( BOX ) E A MOV, 4 CPI, ..XSL4 JRC, ( JUMP IF LESS THAN 4 )
52|( BOX ) B DAD, L BX.X Y STX, H BX.X 1+ Y STX, ( UPDATE )
53|( BOX ) LABEL ..PLOP PIXVAL LDA, M XRA, C ANA, ( PLOP LOOP )
54|( BOX )
55|( 16 BIT INTEGER DIVIDE ROUTINE: M N UN/ Q R ) DECIMAL
56|( SNAP COMMAND )
57|( 8 X 10 CHARACTER SET - ROTATED )
58|( MORE CHARS ) C03F , E03F , 700C , 700C , E03F , C03F , ( A )
59|( CHARS ) E01F , F03F , 3020 , 3028 , F01F , E02F , ( Q )
60|( NEW CHARACTER DRAW ROUTINE )
61|( NORMAL BCD ADDITION )
62|( VGS CPOST , SPOST )
63|( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBCD6 )
64|( TWO DIGIT BCD DISPLAY ROUTINE AND BUMPER )
65|( n-processor MUSPCU, this is starting load block ) HEX
66|( MUSIC EQUATES FOR VECTOR OFFSETS ) HEX
67|( MUSIC VARIABLES & IY EQUATES FOR OFFSETS ) HEX
68|( STEREO EFFECTS RAM AND VOLUME CRES.-DECRES. RAM ) HEX
69|( MUSIC VARIABLES FOR COMPUTER MUSIC GENERATOR AND SYNCER ) HEX
70|( MUSIC PROCESSOR COMANDS ) HEX ( data,PORT )
71|( MUSIC PROCESSOR COMANDS cont. ) HEX
72|( MUSIC PROCESSOR COMANDS cont., STEREO STUFF and ABCRND ) HEX
73|( NOTE CONSTANTS ) HEX
74|( SIN BTABLE FOR LEFT-RIGHT PAN VOLTAGES ) DECIMAL
75|( HELPING SUBR's for MUSPCU *** NOTICE *** ) HEX
76|( HELPING SUBR's cont. ) HEX
77|( MUSIC PROCESSOR- emusic )
78|( OPCODE SUBR's, 0-4 )
79|( OPCODE SUBR's, 5-6, HL= MUSPC )
80|( OPCODE SUBR's, 8-0B, 10H )
81|( OPCODES 0C-0F )
82|( OPCODES 11-16 I/O PORT OUTPUTS and PAN COUNTING, 1AH ) HEX
83|( STEREO OPCODE 1A, THUMPER 1B, MUSIC GENERATOR 07H ) HEX
84|( OPCODE ADDRESS TABLE and FORWARDS ) HEX
85|( COMPMUSIC's +-disp., 15MOD, NOTABLE and THUMPLOCATION ) HEX
86|( STEREO STUFF, LIMITCOUNTING )
87|( ** MUSPCU ** **STEREO** ) HEX
88|( ** MUSPCU ** **STEREO** ) HEX
89|( MUSPCU NOTETIMER, MOSYNC )
90|( MUSPCU cont. MORAMBLE, LOWMOVER, HIGHMOVER )
91|( MUSPCU cont., MORAMBLE cont., STEPMOVER )
92|( MUSPCU VOLUME MOVING )
93|( MUSPCU STEPMOVER, COMPDURMOVER )
94|( MUSPCU cont., TBMOVER, NOMOVER )
95|(

```

```

96|C** MUSIC INTERRUPTER** ) COMPUTER MUSIC ) HEX
97|C MUSCPU cont., ) RANDOM NOTES )
98|C MUSCPU cont., ) PROCESS the score, )
99|C MUSIC PROCESSOR- ) MUSCPUS PUT TOGETHER ) HEX
100|C MUSIC PROCESSOR- ALL xmusics NEED AN IY LOAD ) HEX
101|C MUSCPU SUBROUTINE CALLS )
102|C MUSIC PROCESSOR- EMUSIC, BMUSIC, ... )
103|C MUSIC PROCESSOR- EZMUSIC, B2MUSIC, ... )
104|C CLEAR ANY PRIORITY ON THE MUSIC PROCESSOR )
105|C JAYS VIDEO GAME GOODIES )
106|C QUEUE - VECTOR MANIPULATION ROUTINES )
107|C VECTOR FIELD EQUATES CONTINUED )
108|C STATUS BIT EQUATES )
109|C VGS ) VWRITE )
110|C VGS ) VERASE )
111|C GLOBAL GAME RAM AREA START )
112|C NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
113|C STORAGE ALLOCATOR GOODIES )
114|C ADD NODE TO QUEUE ROUTINE )
115|C DELETE FROM QUEUE )
116|C ADVANCE TO NEXT NODE ON QUEUE )
117|C INCREMENT TIME BASES - C = TIME BASE, IY = Q HEAD )
118|C NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
119|C RESUME BACKGROUND - END INTERRUPT )
120|C TRY TO RUN SOMETHING IN FOREGROUND )
121|C BACKGROUND END INTERRUPT )
122|C BACKGROUND END INTERRUPT )
123|C INTERRUPT START ROUTINE ) HEX
124|C ROUTINE TO DELETE VECTOR IF STATUS SO INDICATES )
125|C MACROS TO GENERATE ANIMATION OPCODES ) DECIMAL
126|C MORE ANIMATION MACRO STUFF )
127|C ANIMATION INTERPRETER ROUTINES )
128|C MORE ANIMATION INTERPRETER ROUTINES )
129|C YET MORE ANIMATION INTERPRETER ROUTINES )
130|C THE ABSOLUTELY LAST SCREEN OF ANIMATION INTERPRETER STUFF )
131|C JUMP TABLE FOR INTERPRETER ROUTINES )
132|C ANIMATION UPDATOR ROUTINE )
133|C DECREMENT ANIMATION TIMERS, COMPUTE VECTORING TIME )
134|C TIME BASED VECTOR UPDATE - IX=VECTOR ADDR, IY=QUEUE ENTRY )
135|C INITIALIZE INTERRUPT VERBS )
136|C SUBROUTINE TO UPDATE PATTERN USING XOR )
137|C SUBROUTINE TO VECTOR USING SECOND DERIVATIVE )
138|C SUBROUTINE TO UPDATE PATTERN USING XOR AND 2ND DERV VECTOR )
139|C UPDATE VECTOR FROM JOYSTICK ) HEX 11 C= JOYSTICK
140|C SUBROUTINE TO UPDATE PATTERN FROM JOYSTICK )
141|C COMPUTE DELTA FOR 1 COORDINATE )
142|C CLEAR VECTOR ) F= INIZL
143|C SUBROUTINE TO PUT VECTOR ON PROCESS Q )
144|C XVMOVE COMMAND - MOVE AN EXISTING VECTOR )
145|C XSTART COMMAND - START AN EXISTING VECTOR )
146|C START A VECTOR WITH JUST INITIAL X AND Y ) DECIMAL
147|C CHECK FOR INTERCEPT WITH VECTOR )
148|C CHECK GROUP OF VECTORS FOR INTERCEPT )
149|C NUMBER PATTERNS , 5 X 7 ORDERED 0-9 )
150|C ROUTINE TO DISPLAY A BCD NUMBER 3 DIGITS LONG FROM VECTOR )
151|C INTERRUPT WRITE NUMBER ROUTINE )
152|C BASE STATION )
153|C SMALL BASE ) DECIMAL DATA SMALBASE 4 B, 11 B, QUAD
154|C GORF ) DECIMAL DATA GORF 6 B, 15 B, QUAD
155|C GORFB ) DECIMAL DATA GORFB 6 B, 15 B, QUAD
156|C GORF 2 AND GORF 3 )
157|C GORF 1 AND GORF 4 )
158|C GORF 5 )
159|C FIRE BASE EXPLOSION PATTERN )
160|C ANOTHER FIREBASE EXPLOSION PATTERN )
161|C CONTINUATION OF FBEXP2, PHASOR AND NULPAT )

```

162|(FBEXP3)
163|(CONTINUED FBEXP3)
164|(FBEXP4)
165|(FBEXP4 CONTINUED)
166|(FIREBASE EXPLOSION 5)
167|(FBEXP5 CONTINUED)
168|(FIRE BASE EXPLOSION 6)
169|(FIRE BASE EXPLOSION 6 CONTINUED)
170|(ALIEN EXPLOSION PATTERN)
171|(MORE ALIEN EXPLOSIONS)
172|(EXPLOSION PATTERNS) DECIMAL
173|(KAMIZAKE PATTERN)
174|(ROTATED KAMIKAZE 1)
175|(ROTATED KAMIKAZE 2)
176|(ROTATED KAMIKAZE 3)
177|(ROTATED KAMIKAZE 4)
178|(MISSIONS- PLAYER'S SHIP EXPLOSION, 1G, SHOOTING SOUND, 1D)
179|(MISSIONS- ZPIP & PZIP SOUNDS- ZP,PZ) HEX
180|(SPACE MISSIONS BMUSIC BLOCK)
181|(MISSIONS- TAKE-OFF- TO) HEX
182|(MISSIONS- DIVE SOUND) HEX
183|(DRAW FIREBASES ON SCREEN)
184|(GAME VARIABLES AND CONSTANTS)
185|(INITIALIZE GAME SCREEN) HEX
186|(RACK UPDATOR)
187|(RADIAL LINE GENERATOR)
188|(RADIAL EFFECT VARIABLES)
189|(NEAT SUBROUTINES)
190|(SUBR TO WRITE NEXT PIXEL IN A LINE)
191|(OTHER NEAT VERBS)
192|(GENERATE A LINE)
193|(LINE GENERATOR - CLIP CHECK)
194|(LINE GENERATOR - SET DELTAS)
195|(ADJUST DELTAS TO QUADRANT, AND BIAS TO EFFECT CENTER)
196|(WRITE ONLY ENTRY AND SET CENTER OF LINE EFFECT)
197|(SHIFT RIGHT ARITHMETIC BY N ROUTINE)
198|(SPIRAL VECTOR ROUTINE)
199|(INTERRUPT ROUTINE TO SPIRAL VECTOR)
200|(SUBROUTINES TO CALCULATE DISPLACEMENTS FOR RACK MEMBER) HEX
201|(WAIT AND ANIMATION TRACKING TABLE ROUTINES) HEX
202|(RECOMPUTE LIMITS) HEX
203|(SUBR TO STEP MASTER COORDS ONE TICK AND LIMIT CHECK) HEX
204|(WE FOUND AN INVADER - WRITE HIM)
205|(REWRITE A RACK MEMBER USING NORMAL PATTERNS)
206|(REENTER RACK) HEX
207|(INTERRUPT ROUTINE TO REENTER A GALAXIAN) DECIMAL
208|(CHECK FOR INTERCEPT WITH RACK MEMBER)
209|(ANIMATION LIST AND ROUTINE TO EXPLODE THE FIREBASE)
210|(SCORIN) HEX TABLE ASTBL 60 , 60 , 80 , 100 , 300 , 200 ,
211|(MORE SCORING GOODIES)
212|(BACKGROUND PHASOR INTERCEPT PROCESSING ROUTINES)
213|(ROUTINE TO CALL FROM SCAN LOOP)
214|(ANIMATION SUBR TO INITIALIZE THE FIRE BASE)
215|(EXPLODE THE FINAL FIREBASE SOMEWHAT MORE SPECTACULAR)
216|(CHECK FOR PLAYER HIT)
217|(COMMON INITIALIZATION GOODIES)
218|(SPECIAL ROUTINE TO MOVE PHASOR BLAST)
219|(START OR RESTART THE PHASOR MOVING)
220|(CHECK FIRE SWITCH)
221|(AWAIT THE ARRIVAL OF THE VERTICAL INTERVAL)
222|(NEW COLOR ROUTINES)
223|(FADE UP/DOWN ROUTINES)
224|(FORCE FIELD DRAWER) DECIMAL
225|(MORE FORCE FIELD GOODIES)
226|(CHECK FOR INTERCEPT WITH ANY OF THE ATTACKERS)
227|(POSITION OBJECT RELATIVE TO FORMATION LEADER)

228|C INTERRUPT ROUTINE TO WRITE RELATIVE FORMATION MEMBER)
229|C LEADER X Y ANIMATION TIME STATUS VECTOR FSTART)
230|C EFFECT REENTRY INTO RACK OR FORMATION)
231|C INTERRUPT ROUTINE TO REENTER KAMIKAZE)
232|C ROUTINE TO RETARGET AN ATTACKER)
233|C ROUTINE TO FLIP OVER ATTACKER)
234|C LEFT ROLL SEQUENCE)
235|C RIGHT ROLL SEQUENCE)
236|C KAMIKAZE ATTACK ANIMATION)
237|C ANIMATION TO ACTIVATE KAMIKAZES)

```

+-----Block      30-----
0|( ( CONSTANTS AND IO PORT DEFINITIONS )
1|( : C= ) CONSTANT ( ; ) ( : V= ) VARIABLE ( ; )
2|HEX 0 C= CC? ( 0 TO CROSS COMPILE, 1 FOR NORMAL )
3|CC? IFTRUE 0F002 C= RAMBASE 0FFFF C= LASTRAMADDR
4|OTHERWISE 0D000 C= RAMBASE 0DFFF C= LASTRAMADDR IFEND
5|( : NC= ) 1+ DUP C= ( ; ) ( : SC= ) DUP C= ( ; )
6|( : T= ) TABLE ( ; ) ( : A= ) ARRAY ( ; )
7|( : BT= ) BTABLE ( ; ) ( : BA= ) BARRAY ( ; )
8|( : BV= ) BVARIABLE ( ; ) ( : F= ) FORWARD ( ; )
9|0 C= COL0R 1 C= COL1R 2 C= COL2R 3 C= COL3R
10|4 C= COL0L 5 C= COL1L 6 C= COL2L 7 C= COL3L
11|0B C= COLBX 9 C= HORCB 0A C= VERBL 7F C= STARZ
12|10 C= TONMO 11 C= TONEA 12 C= TONEB 13 C= TONEC
13|14 C= VIBRA 16 C= VOLAB 15 C= VOLC 17 C= VOLN 18 C= SNDBX
14|0D C= INFBK 0E C= INMOD 0F C= INLIN 8 C= CONCM 0F C= HORAF
15|0C C= MAGIC 19 C= XPAND 8 C= INTST 0E C= VERAF -->

```

```

+-----Block      31-----
0|( ( RAM ORIGIN AND SPECIAL VGS VERBS )
1|RAMBASE VPTR ! ( START RAM AT RAMBASE! )
2|
3|CODE DI DI, NEXT ( disable interrupts )
4|CODE EI EI, NEXT ( enable interrupts )
5|CODE XDI DI, A XRA, INMOD OUT, NEXT
6|: MS 0 DO 4 0 DO LOOP LOOP ;
7|CC? IFTRUE
8|: ROMIT DP @ other @ DP ! ; : TIMOR DP @ other ! DP ! ;
9|: <ONSCR ASM DEFINITIONS DP @ there @ DP ! ;
10|: ONSCR> DP @ there ! DP ! TERSE DEFINITIONS ;
11|ROMIT
12|IFEND
13|-->
14|
15|

```

```

+-----Block      32-----
0|( VGS random, ranger )
1|2 A= RND# ( you must seed RND# !!!!!!!! )
2|SUBR random ( 32 bit random # generator )
3| ( out- randomly selected # in DEHL )
4| B PUSH, 0 RND# LBCD, 1321 H LXI, B DAD, H PUSH,
5| 2776 H LXI, B DADC, 1 RND# LDED, D DAD, XTHL,
6| B DAD, XTHL, D DADC, XTHL, B DAD, XTHL, D DADC, XTHL,
7| E D MOV, 3 E MOV, C B MOV, 0 C MVI, B DAD, 0 RND# SHLD,
8| XTHL, D DADC, 1 RND# SHLD, D POP, B POP, RET,
9|SUBR ranger ( pass 0 in HL, range in DE, ) B PUSH, EXX,
10| 0 H LXI, H D MOV, L E MOV, EXX, D PUSH, B POP, XCHG,
11| 0 H LXI, BEGIN, 3 CLAR, C RARR, CY, IF, B DAD, EXX, D DADC,
12| EXX, THEN, B A MOV, C ORA, 1, IF, E CLAR, D RALR, EXX, E RALR,
13| D RALR, EXX, ( SWAP ) JMP, THEN, EXX, B POP, RET,
14|-->
15|

```

```

+-----Block      33-----
0|( VGS          random, RND, RANGER, RANGERND )
1|SUBR rnd.( pass range in DE, returns # in HL )
2| D PUSH, random CALL, D POP, ranger CALL, RET,
3|CODE RANDOM ( out= # on stack ) random CALL, H PUSH, NEXT
4|CODE RANGER ( pass range, # on stack )
5| ( result is # times range / FFFF, ie. 30H 8000H ---- is 18H )
6| D POP, H POP, ranger CALL, H PUSH, NEXT
7|CODE RND ( pass range on stack )
8| random CALL, D POP, ranger CALL, H PUSH, NEXT
9|-->
10|
11|
12|
13|
14|
15|
+-----Block      34-----
0|( UPDATE COLOR MAP -- QUICKLY )
1|CODE COLOR EXX, H POP, 800 B LXI,
2|BEGIN, M A MOV, A OUTP, H INX, C INR, LOOP,
3|EXX, NEXT
4|-->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      35-----
0|( VGS          FLOOD )
1|CODE FLOOD ( set all color ports to the same value )
2| ( in- byte color value )
3| ( out- screen color ports set to same value )
4|EXX, H POP, L A MOV, 800 B LXI, BEGIN, A OUTP, C INR, LOOP,
5|EXX, NEXT
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block 36-----
0| ( VGS          FILL )
1|: FILL ( fill screen with constant data )
2| ( in- constant , starting address , # of bytes to fill )
3| ( out- does sequential fill with constant specified )
4| ROT ROT 2DUP ! SWAP DROP DUP 1+ ROT 1- BMOVE ;
5| DECIMAL -->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block 37-----
0| ( VGS write routines pattern representation )
1| -->          PATTERN REPRESENTATION
2| Pattern header requirements are determined by the write
3| routine used. The following diagram shows the hierarchy
4| used :
5|
6|          WRITEP          X size
7|          WRITE          Y size
8|          WRITE          pattern data ---
9|
10| If a pattern is to be written with a shift it must be self
11| flushing. A pattern is self flushing if the right side 3
12| bits are all zero. Pattern padding is required in some cases.
13|
14|
15|

```

```

+-----Block 38-----
0| ( VGS          pattern board and magic equates )
1| HEX
2| ( pattern board ports )
3| 78 C= PBLINADRL 79 C= PBLINADRH 7A C= PBSTAT 7B C= PBAREADRL
4| 7C C= PBXMOD 7D C= PBAREADRH 7E C= PBXWIDE 7F C= PBYHIGH
5| ( pattern board status port bits )
6| 0 C= PBDIR 1 C= PBEXP 2 C= PBCONS 3 C= PBFLUSH
7| 4 C= PBFLIP 5 C= PBFLIP
8| ( magic register bits )
9| 2 C= MRROT 3 C= MREXP 4 C= MROR 5 C= MRXOR
10| 6 C= MRFLIP 7 C= MRFLIP
11| -->
12|
13|
14|
15|

```

```

+-----Block      39-----
0|( RELABS ) F= relabs SUBR ffre labs <ASSEMBLE
1|7 C BIT, 0<>, IF, 1 Y A LDX, H ADD, A DCR, A H MOV,
2|THEN, 6 C BIT, 0<>, IF, 0 Y A LDX, D ADD, A DCR,
3|A D MOV, THEN, ( FALL INTO ... )
4|LABEL relabs ( relative X Y to magic address conversion )
5| ( in- BC=exp/mag DE=x HL=y )
6| ( out- BC=exp/mag+shift HL=schrdr )
7| H A MOV, 0 H MVI, A L MOV,
8| H DAD, H DAD, H DAD,
9| H DAD, D PUSH, L E MOV, H D MOV, H DAD, H DAD, ( *64 )
10| D DAD, ( *80 ) XCHG, H POP, ( x )
11| L A MOV, ( SAVE BIT CNT ) H L MOV, 0 H MVI, D DAD, ( x+y )
12| RLC, RLC, HEX 3 ANI,
13| MRFLOP C BIT, 0<>, IF, NEG, 0=, IF, H DCX, THEN, THEN,
14| 3 ANI, A E MOV, C A MOV, FC ANI, E ORA, A C MOV, RET,
15|ASSEMBLE> -->

```

```

+-----Block      40-----
0|( VGS      RELABS )
1|CODE RELABS ( relative to absolute conversion )
2| ( in- exp/mag , X , Y )
3| ( out- exp/mag+shift , schrdr )
4| EXX, ( save BC )
5| H POP, ( Y ) D POP, ( X ) B POP, ( exp/mag )
6| relabs CALL, B PUSH, ( exp/mag+shf )
7| H PUSH, ( schrdr ) EXX, NEXT
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      41-----
0|( VGS      write )
1|SUBR write ( write pattern on screen )
2| ( in- BC=exp/mag+shift DE=y/x size HL=schrdr IY=patadr )
3| ( WRTSYS 0<> for pattern board 0= for software write )
4| ( out- C=mag+shift ; pattern on screen )
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      42-----
0| ( VGS                      write con't. )
1| ( pattern board write )
2| B A MOV, XPAND OUT, C A MOV, MAGIC OUT, HEX 24 A MVI,
3| MRFLIP C BIT, 0<>, IF, PBFLIP A SET, THEN,
4| MRFLOP C BIT, 0<>, IF, PBFLOP A RES, THEN,
5| MREXP C BIT, 0<>, IF, PBEXP A SET, THEN,
6| A B MOV, PBSTAT OUT, ( B=status C=magic )
7| H PUSH,
8| Y PUSHX, H POP, L A MOV, PBLINADRL OUT,
9|                      H A MOV, PBLINADRH OUT,
10| H POP,
11|                      L A MOV, PBAREADRL OUT,
12|                      H A MOV, PBAREADRH OUT,
13|-->
14|
15|
+-----Block      43-----
0| ( VGS                      write con't. )
1| E H MOV, ( X size )
2| MREXP C BIT, 0<>, IF, H RLCR, ( *2 ) THEN,
3| H DCR, ( H=X size zero relative )
4| MRFLIP C BIT, 0<>, IF,
5|     MRFLOP C BIT, 0<>, IF, DECIMAL -80 A MVI, H ADD,
6|     ELSE, DECIMAL -80 A MVI, H SUB, THEN,
7|     ELSE,
8|     MRFLOP C BIT, 0<>, IF, DECIMAL 80 A MVI, H ADD,
9|     ELSE, DECIMAL 80 A MVI, H SUB, THEN,
10|    THEN, ( A=Xmod ) PBXMOD OUT,
11| HEX H A MOV, PBXWIDE OUT,
12| D A MOV, ( Y size ) A DCR, ( 0 rel ) PBYHIGH OUT,
13| RET,
14|-->
15|
+-----Block      44-----
0| ( VGS                      writep )
1| SUBR writep ( does write with pattern size header on pattern )
2| ( in- BC=exp/mag+shift DE=y/x size HL=scradr IY=patadr )
3| ( WRTSYS 0<> for pattern board 0= for software write )
4| ( out- C=mag+shift ; pattern on screen )
5| 0 Y E LDX, ( X size ) Y INXX,
6| 0 Y D LDX, ( Y size ) Y INXX, write JMP,
7|-->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      45-----
0|( VGS                WRITEP )
1|CODE WRITEP  ( write with pattern size header on pattern )
2|  ( in- x , y , patadr , ex/mag )
3|      ( WRTSYS 0<> for pattern board 0= for software write )
4|  ( out- pattern on screen )
5|  Y PUSHX, H POP, EXX, B POP, Y POPX, H POP, D POP,
6|  relabs CALL,
7|  writep CALL, EXX, H PUSH, Y POPX, NEXT
8|CODE FFWRITEP Y PUSHX, H POP, EXX, B POP, Y POPX, H POP, D POP,
9|ffrelabs CALL, writep CALL, EXX, H PUSH, Y POPX, NEXT
10|-->
11|
12|
13|
14|
15|

+-----Block      46-----
0|( VGS                WRITE )
1|CODE WRITE  ( write with X Y sizes ; pattern with no header )
2|  ( in- x , y , patadr , y/x size ex/mag )
3|      ( WRTSYS 0<> for pattern board 0= for software write )
4|  ( out- pattern on screen )
5|  Y PUSHX, H POP, EXX, B POP, ( ex/mag ) H POP, ( sizes )
6|  Y POPX, ( patadr ) D POP, ( Y ) XTHL, ( H<-X S<-sizes )
7|  XCHG, ( X<->Y ) relabs CALL, D POP, ( sizes )
8|  write CALL, EXX, H PUSH, Y POPX, NEXT
9|-->
10|
11|
12|
13|
14|
15|

+-----Block      47-----
0|( FRAME, UNFRAME MACROS )
1|2 C= FR.P1 4 C= FR.P2 6 C= FR.P3
2|{ : FRAME } { [ ] ASM { ] } Y PUSHX, 0 Y LXIX, SP DADY, { ; }
3|{ : UNFRAME } { [ ] ASM { ] } Y POPX, { ; }
4|-->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      48-----
0|( SPECIAL RELABS FOR BOX )
1|SUBR R2A ( RELATIVE TO ABSOLUTE CONVERSION FOR BOX AND LINE )
2|E A MOV, 3 ANI, PSW PUSH, D PUSH, D SRLR, E A MOV, RAR,
3|A ANA, RAR, C L MOV, 0 H MVI, H DAD, H DAD, H DAD,
4|H DAD, L E MOV, H D MOV, H DAD, H DAD, D DAD,
5|A E MOV, 0 D MVI, D DAD, D POP, PSW POP, RET,
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      49-----
0|( PATTERN BOARD BOX COMMAND )
1|( X Y XS YS MODE BOX )
2|( PARAMETER ADDRESS EQUATES )
3|DECIMAL
4|2 C= BX.M 4 C= BX.YS 6 C= BX.XS 8 C= BX.Y 10 C= BX.X
5|( SCRATCH AREA USED BY BOX COMMAND )
6|0 B VARIABLE WRMODE 0 B VARIABLE PIXVAL
7|( F=ARD REFERENCE DECLARATIONS )
8|F= ..SKIP F= ..MNZ F= ..XSL4 F= ..SLOB F= PBBOX
9|F= ..XST1 F= ..STRC F= ..XSTF F= ..XSBG F= ..BOXP
10|F= ..PLOP F= OUTPB F= ..CPBB F= ..FFF F= ..DOSF
11|F= ..XSLA F= ..XORL F= ..OUTM F= NULRET
12|HEX DATA MSKTBL 0 B, 55 B, AA B, FF B,
13|-->
14|
15|

+-----Block      50-----
0|( X Y XS YS MODE BOX ) HEX
1|CODE BOX <ASSEMBLE
2|      FRAME   EXX,
3|      BX.M Y A LDX,
4|      A C MOV,      ( IS MODE = 4 )
5|      4 CPI, ..SKIP JZ,      ( IF SO SKIP IT )
6|      8 CPI, ..SKIP JNC,      ( SKIP IF >= 8 TOO )
7|      4 ANI, WRMODE STA,      ( ISOLATE AND STUFF MODE )
8|      C A MOV, 0 ANI, A C MOV, ( GET A BYTE ALL THE SAME )
9|      0 B MVI, MSKTBL H LXI,
10|     B DAD, M A MOV, PIXVAL STA, ( AND REMEMBER AS PIXVAL )
11|     BX.XS Y E LDX, BX.XS 1+ Y D LDX, ( DE=XS )
12|LABEL ..BOXP E A MOV, D ORA, ..SKIP JRZ, ( QUIT IF XS=0 )
13|     BX.X Y A LDX, 3 ANI,      ( ON A BYTE BOUNDARY? )
14|     ..MNZ JRNZ,      ( NO - JUMP )
15|     D A MOV, A ORA, ..CPBB JRNZ, ( IF >256 USE PB ) -->

```



```

+-----Block      51-----
0| ( BOX ) E A MOV, 4 CPI, ..XSL4 JRC,      ( JUMP IF LESS THAN 4 )
1|      8 CPI, ..SLOB JRC,      ( IF <8 DON'T USE PB )
2| LABEL ..CPBB PBBOX CALL, ..XST1 JMPR,    ( CALL DRAW WITH PB )
3| LABEL ..SLOB FF C MVI, ..STRC CALL,      ( PAINT A FULL STRIPE )
4|      4 C MVI, ..XSTF JMPR,
5| LABEL ..MNZ 3 ANI, A B MOV, ( COMPUTE MIN(X,4-MOD(XS-4 )
6|      4 A MVI, B SUB, D 0 BIT, ..XSBG JNZ, ( JUMP IF XS>256 )
7|      E CMP, ..XSBG JC, E A MOV,      ( OR > MOD )
8| LABEL ..XSBG A C MOV, B PUSH, B C MOV, ( MOD IS BIGGER )
9|      A B MOV, A XRA,
10| LABEL ..FFF RRC, RRC, C0 ORI, ..FFF DJNZ, C B MOV, ( MASK )
11| LABEL ..DOSF RRC, RRC, 3F ANI, ..DOSF DJNZ, ( SHIFT MOD TIMES )
12|      A C MOV, ..STRC CALL, B POP,      ( DRAW PART STRIPE )
13| LABEL ..XSTF 0 B MVI,
14| LABEL ..XST1 BX.X Y L LDX, BX.X 1+ Y H LDX, ( HL=X )
15|-->

```

```

+-----Block      52-----
0| ( BOX ) B DAD, L BX.X Y STX, H BX.X 1+ Y STX, ( UPDATE )
1| XCHG, A XRA, B DSBC, XCHG, ..BOXP JMP, ( SUBTRACT SIZE )
2| LABEL ..XSL4 A B MOV, A XRA,      ( PAINT FINAL STRIPE )
3| LABEL ..XSLA RRC, RRC, C0 ORI, ..XSLA DJNZ, ( FORM FINAL MSK )
4|      A C MOV, ..STRC CALL,      ( AND DO FINAL STRIPE )
5| LABEL ..SKIP UNFRAME H POP, H POP, H POP, H POP, H POP,
6|      EXX, NEXT ( QUIT CIRCLE ROUTINE )
7| LABEL ..STRC D PUSH, C B MOV, BX.X Y E LDX, ( STRIPE DRAWER )
8|      BX.X 1+ Y D LDX, BX.Y Y C LDX, ( GET COORDS )
9|      R2A CALL, H 6 SET, B C MOV,      ( R2A, UNMAGIC, RESET )
10|     50 D LXI, BX.YS Y B LDX, WRMODE LDA, A ANA,
11|     ..PLOP JRZ,      ( JUMP IF PLOP, XOR ROUTINE FOLLOWS )
12| LABEL ..XORL PIXVAL LDA, C ANA, M XRA, A M MOV, ( UPDATE PIXL )
13|     D DAD, ..XORL DJNZ,      ( UPDATE ADDR, LOOP BACK )
14|     D POP, RET,      ( XOR STRIPE DONE )
15|-->

```

```

+-----Block      53-----
0| ( BOX ) LABEL ..PLOP PIXVAL LDA, M XRA, C ANA, ( PLOP LOOP )
1|      M XRA, A M MOV, D DAD, ..PLOP DJNZ, ( USE XOR TRICK )
2|      D POP, RET,
3| LABEL PBBOX
4|      D PUSH, D SRLR, E RARR, D SRLR, E RARR, ( DE=DE/4 )
5|      E B MOV,
6|      BX.Y Y C LDX, BX.X Y E LDX, BX.X 1+ Y D LDX, ( COORDS )
7|      R2A CALL, WRMODE LDA, 4 ANI, ( CONVERT, CHECK WR TYPE )
8|      ..OUTM JRZ, 20 A MVI, ( JUMP IF PLOP, ELSE ITS XOR )
9| LABEL ..OUTM MAGIC OUT, 20 A MVI, PIXVAL D LXI, ( SET MT, ST )
10|     OUTFB CALL, B DCR, 50 A MVI, B SUB, ( COMPUTE XYOD )
11|     PBXMOD OUT, B A MOV, PBXWIDE OUT, ( THEN WIDTH )
12|     BX.YS Y A LDX, A DCR, PBYHIGH OUT, ( THEN HEIGHT )
13|     D POP, B L MOV, 0 H MVI, L INR, ( COMPUTE LINES USED )
14|     H DAD, H DAD, L C MOV, H B MOV, RET,
15|-->

```

```

+-----Block      54-----
0|( BOX )
1|LABEL OUTPB ( ROUTINE TO OUTPUT STUFF TO PAT BOARD )
2|      PBSTAT OUT, E A MOV, PBLINADRL OUT, ( STAT AND LINEAR )
3|      D A MOV, PBLINADRH OUT, L A MOV,
4|      PBAREADRL OUT, H A MOV, PBAREADRH OUT, ( AREA )
5|LABEL NULRET RET,
6|ASSEMBLE>
7|DECIMAL
8|2 +BLOCK CONTINUED
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      56-----
0|( 16 BIT INTEGER DIVIDE ROUTINE: M N UN/ Q R ) DECIMAL
1|FORWARD .ZERO FORWARD IDV50 FORWARD IDV60
2|FORWARD IDV10 FORWARD IDV20 FORWARD IDV30 FORWARD IDV40
3|SUBR unsdiv <ASSEMBLE L C MOV, H B MOV, D A MOV, 0 H LXI,
4|E ORA, .ZERO JRZ, B A MOV, 16 B MVI,
5|LABEL IDV10 C RALR, RAL, H DADC, D DSBC,
6|LABEL IDV20 CMC, IDV50 JRNC,
7|LABEL IDV30 IDV10 DJNZ, IDV60 JMPR,
8|LABEL IDV40 C RALR, RAL, H DADC, A ANA, D DADC,
9|IDV30 JRC, IDV20 JRZ,
10|LABEL IDV50 IDV40 DJNZ, D DAD, A ANA, ( MAKE IT POS )
11|LABEL IDV60 C RALR, RAL, A D MOV, C E MOV,
12|LABEL .ZERO RET, ASSEMBLE>
13|SUBR UNSDIV H PUSH, D DSBC, CY, IF, 0 D LXI, H POP, ELSE,
14|H POP, unsdiv CALL, THEN, RET, CODE UN/ EXX, D POP, H POP,
15|UNSDIV CALL, H PUSH, D PUSH, EXX, NEXT DECIMAL -->

```

```

+-----Block      57-----
0|( SNAP COMMAND )
1|HEX CODE snap EXX,
2|25 A MVI, PBSTAT OUT,
3|H POP,
4|D POP, E A MOV, PBAREADRL OUT, D A MOV, 40 ORI, PBAREADRH OUT,
5|D POP, B POP,
6|B INX, B INX, B INX, B SRLR, C RARR, B SRLR, C RARR,
7|C A MOV, PBXWIDE OUT, A INR, A M MOV, H INX, E M MOV, H INX,
8|L A MOV, PBLINADRL OUT, H A MOV, PBLINADRH OUT,
9|50 A MVI, C SUB, PBXMOD OUT,
10|E A MOV, A DCR, PBXHIGH OUT, ( DO IT TO IT )
11|EXX, NEXT
12|: SNAP 0 ROT ROT RELABS SWAP DROP SWAP snap ;
13|DECIMAL -->
14|
15|

```

```

+-----Block      58-----
0|( 8 X 10 CHARACTER SET - ROTATED )
1|HEX
2|DATA CHRTBL
3|0000 , 0000 , 0000 , 0000 , 0000 , 0000 , ( SPACE )
4|E01F , F03F , 3030 , 3030 , F03F , E01F , ( 0 )
5|0000 , 2030 , D03F , F03F , 0030 , 0000 , ( 1 )
6|603E , 703F , 3033 , 3033 , F033 , E031 , ( 2 )
7|6018 , 7038 , 3033 , 3033 , F03F , E01E , ( 3 )
8|F003 , F003 , 0003 , 0003 , F03F , F03F , ( 4 )
9|F01B , F03B , 3033 , 3033 , 303F , 001E , ( 5 )
10|E01F , F03F , 3033 , 3033 , 303F , 001E , ( 6 )
11|3000 , 3038 , 303E , B00F , F003 , F000 , ( 7 )
12|E01E , F03F , 3033 , 3033 , F03F , E01E , ( 8 )
13|E001 , F003 , 3033 , 3033 , F03F , E01F , ( 9 )
14|-->
15|

```

```

+-----Block      59-----
0|( MORE CHARS ) C03F , E03F , 700C , 700C , E03F , C03F , ( A )
1|F03F , F03F , 3033 , 3033 , F03F , E01E , ( B )
2|E01F , F03F , 3030 , 3030 , 7038 , 6018 , ( C )
3|F03F , F03F , 3030 , 3030 , F03F , E01F , ( D )
4|F03F , F03F , 3033 , 3033 , 3033 , 3030 , ( E )
5|F03F , F03F , 3003 , 3003 , 3003 , 3000 , ( F )
6|E01F , F03F , 3030 , 3036 , 303E , 201E , ( G )
7|F03F , F03F , 0003 , 0003 , F03F , F03F , ( H )
8|0000 , 3030 , F03F , F03F , 3030 , 0000 , ( I )
9|001C , 003C , 0030 , 0030 , F03F , F01F , ( J )
10|F03F , F03F , 8003 , C00F , F03C , 7038 , ( K )
11|F03F , F03F , 0030 , 0030 , 0030 , 0030 , ( L )
12|F03F , 6000 , 8001 , 8001 , 6000 , F03F , ( M )
13|F03F , F03F , C001 , 8003 , F03F , F03F , ( N )
14|E01F , F03F , 3030 , 3030 , F03F , E01F , ( O )
15|F03F , F03F , 3003 , 3003 , F003 , E001 , ( P ) -->

```

```

+-----Block      60-----
0|( CHARS ) E01F , F03F , 3020 , 3028 , F01F , E02F , ( Q )
1|F03F , F03F , 3003 , 3007 , F03F , E03D , ( R )
2|E019 , F03B , 3033 , 3033 , 703F , 601E , ( S )
3|3000 , 3000 , F03F , F03F , 3000 , 3000 , ( T )
4|F01F , F03F , 0030 , 0030 , F03F , F01F , ( U )
5|F003 , F00F , 003E , 003E , F00F , F003 , ( V )
6|F03F , 000C , 8007 , 8007 , 000C , F03F , ( W )
7|7038 , F03C , C00F , C00F , F03C , 7038 , ( X )
8|7000 , F000 , C03F , C03F , F000 , 7000 , ( Y )
9|303C , 303E , 3037 , 303C , F031 , F030 , ( Z )
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      61-----
0|( NEW CHARACTER DRAW ROUTINE )
1|( IN HL=Y DE=X BC=EXPAND/MAGIC A= CHAR TO DISPLAY )
2|HEX
3|SUBR drawchar B PUSH, H PUSH, D PUSH, 20 SUI, 0<>, IF,
4|0F SUI, 0B CPI, CY~, IF, 7 SUI, THEN, THEN,
5|A L MOV, 0 H MVI, H DAD, H DAD, L E MOV, H D MOV,
6|H DAD, D DAD, CHRTBL D LXI, D DAD, H PUSH, Y POPX,
7|D POP, H POP, H PUSH, D PUSH, relabs CALL,
8|602 D LXI, write CALL, D POP, H POP, H A MOV, 7 ADI,
9|A H MOV, B POP, RET,
10|
11|( TERSE INTERFACE - X Y COLOR/MAGIC CHAR cpost --- NEW X Y )
12|CODE cpost EXX, B POP, C A MOV, B POP, H POP, D POP,
13|X PUSHX, Y PUSHX, drawchar CALL, Y POPX, X POPX,
14|D PUSH, H PUSH, B PUSH, EXX, NEXT
15|DECIMAL -->

```

```

+-----Block      62-----
0|( NORMAL BCD ADDITION )
1|CODE BCD+! EXX, H POP, D POP,
2|M A MOV, E ADD, DAA, A M MOV,
3|H INX, M A MOV, D ADC, DAA, A M MOV,
4|H INX, M A MOV, 0 ACI, DAA, A M MOV,
5|EXX, NEXT
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      63-----
0|( VGS                                CPOST , SPOST )
1|: CPOST ( post an ascii-character on the screen ; see options )
2|  ( in= x , y , opt+ex/mag , ascii-char )
3|    ( WRTSYS 0<> for pattern board 0= for software write )
4|  ( out- character on screen )
5|  cpost DROP DROP DROP ;
6|
7|: SPOST ( post an ascii-string on the screen ; see options )
8|  ( in= x , y , opt+ex/mag , addr , count )
9|    ( i.e. 0 0 20 A" STRING" COUNT SPOST )
10|    ( WRTSYS 0<> for pattern board 0= for software write )
11|    ( cannot be used in immediat mode )
12|  ( out- character on screen )
13|  OVER + SWAP DO I B? cpost LOOP DROP DROP DROP ;
14|-->
15|

```

```

+-----Block      64-----
0|( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBCD6 )
1|HEX SUBR digit 0F ANI, 0=, IF, D ORA, 0<>, IF, 0F0 A MVI, THEN,
2|ELSE, 0 D MVI, THEN, 30 ADI, EXX, drawchar CALL, EXX, RET,
3|HEX
4|
5|F= DGTL
6|CODE DISPBCD6 <ASSEMBLE H POP, M A MOV, H INX, M ORA,
7|H INX, M ORA, A D MOV, 3 E MVI,
8|EXX, B POP, H POP, D POP, X PUSHX, Y PUSHX, EXX,
9|LABEL DGTL M A MOV, RRC, RRC, RRC, RRC, digit CALL,
10|M A MOV, digit CALL, H DCX, E DCR, DGTL JRNZ,
11|Y POPX, X POPX, NEXT <ASSEMBLE>
12|CC? IFTRUE TIMOR IFEND
13|DECIMAL -->
14|
15|
+-----Block      65-----
0|( TWO DIGIT BCD DISPLAY ROUTINE AND BUMPER )
1|CODE DISPBCD2 H POP, EXX, B POP, H POP, D POP,
2|X PUSHX, Y PUSHX, EXX,
3|1 D MVI, L A MOV, RRC, RRC, RRC, RRC, digit CALL,
4|0 D MVI, L A MOV, digit CALL,
5|Y POPX, X POPX, NEXT
6|
7|CODE BCDBUMP H POP, M A MOV, 1 ADI, DAA, A M MOV, NEXT
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|
+-----Block      66-----
0|( n-processor MUSPCU, this is starting load block ) HEX
1|( old TERSE music still works and runs on this MUSCPU or )
2|( 2MUSCPU, multiple processors reload IY for music vector, )
3|( and the variable SOUNDBOX to the port 1 past the NOISE port, )
4|( ie. sounds are in 10-17, set SOUNDBOX to 18H. )
5|0 BV= MUSICFLAG ( turns off all processors for GAMEOVER )
6|0 BV= THUMPCOUNTER ( SPECIAL SPACE MISSIONS OPCODE )
7|CC? IFTRUE <ONSCR IFEND
8|18 C= CHIP1 58 C= CHIP2 ( high port of soundbox for any chips )
9|( EMUSIC uses CHIP1, E2MUSIC uses CHIP2, ... add EMUSIC's )
10|6E C= PANPORT1 6C C= PANPORT2 ( left-low port of stereo pair )
11|( TASK EQUATES FOR VECTOR OFFSETS ) HEX
12|( : { ( ) ( ; ) ( : ) } ( ; ) ( for relative RAM order )
13|-->
14|
15|

```

```

+-----Block      67-----
0|( MUSIC EQUATES FOR VECTOR OFFSETS ) HEX
1|({ 0 SC= BEGMUSRAM ( first byte of music-vector )
2|SC= MUSPC SC= MUSPCH NC= MUSPCL ( music program counter )
3|NC= STARTPC SC= STARTPCH NC= STARTPCL ( startover address )
4|NC= SOUNDBOX ( highest port of I/O chips sound ports )
5|NC= PANPORT# ( bottom and left port of stereo output )
6|NC= MOVALUE ( currant value )
7|NC= VIBTRACKER ( vibrato convience tracker for games ) }}
8|({ NC= MULTIPLE NC= PRIORITY }} ( for repeated and important )
9|({ NC= RAMPFLAG ( /|/|/ vs. /\|/ ) NC= RAMBLEFLAG ( on/off )
10|NC= HIGHLIM NC= LOWLIM NC= STEP ( pertaining to MO walk )
11|NC= RAMBLETIMER ( master oscillator timer )
12|NC= TIMEBASE ( reload rambletimer value ) }}
13|NC= LIMCOUNTER ( MO limit counter )
14|-->
15|

```

```

+-----Block      68-----
0|( MUSIC VARIABLES & 1Y EQUATES FOR OFFSETS ) HEX
1|({ NC= STOPTB ( stopvalue for timebase-mover )
2|NC= TBSTEP NC= TBTB NC= TBTIMER ( tbmover's ss,tb,timer ) }}
3|({ NC= NOSTOP ( noisemover's sv,ss,timer,tb,tracker )
4|NC= NOSTEP NC= NOTIMER NC= NOTIMEBASE NC= NOVALUE }}
5|({ NC= STOPSTEPS ( MO's stepmover etc. )
6|NC= BIGOFASTEP NC= STEPTIMEBASE NC= STEPTIMER }}
7|({ NC= STOPLOWLIM ( lowlim's mover's ram, stopvalue )
8|NC= LOWSTEP ( ss or stepsize )
9|NC= LOW# ( # of limits to hit before moving )
10|NC= LOWCOUNTER ( counting low# down ) }}
11|({ NC= STOPHIGHLIM ( highmover's ram )
12|NC= HIGHSTEP NC= HIGH# NC= HIGHCOUNTER }}
13|-->
14|
15|

```

```

+-----Block      69-----
0|( STEREO EFFECTS RAM AND VOLUME CRES.-DECRES. RAM ) HEX
1|( total pan volumes either channel, FFH, 64 STEPS BETWEEN )
2|( load lowest port in PANPORT#, this is left side, 1+ right )
3|( watch step starting direction for left-right action )
4|({ NC= LEFTPAN ( tracker )
5|NC= PANSTEP ( step size )
6| ( timebase for updating )
7|NC= PANTIMEBASE NC= PANTIMER
8| ( count # of limits to achieve )
9|NC= PANCOUNTER }}
10|({ NC= VOLHIGHLIM NC= VOLLOWLIM NC= VOLSTEP
11|NC= VOLTIMEBASE NC= VOLTIMER
12|NC= MCTRACKER ( AB volumes taken from C )
13|-->
14|
15|

```

```

+-----Block      70-----
0|( MUSIC VARIABLES FOR COMPUTER MUSIC GENERATOR AND SYNCER ) HEX
1|({ NC= SYNCMO NC= STARTMC ( special bvarbs for THUMPING ) })
2|({ NC= NOTETIMER ( note timer )
3|NC= COMPDURATION ( computer music note duration )
4|NC= COMPSTEP ( step = { 1,0,-1 } )
5|NC= COMPTIMER ( for COMPDURATION moving )
6|NC= ATRACKER NC= BTRACKER NC= CTRACKER NC= MOTRACKER
7|NC= NOTECOUNTER ( for key changes ) })
8|( trackers of indecies to NOTABLE and MOTABLE )
9|NC= MST ( MUSIC-STATE-TRANSITION jump around variable )
10|NC= ENDMUSRAM ( last byte of ram )
11|80 C= COMPTB CC? IFTRUE SWAP ONSCR> IFEND
12|DUP BARRAY MUSIC-BARRAY-1
13|BARRAY MUSIC-BARRAY-2 CC? IFTRUE <ONSCR IFEND
14|{ : MB1 } 0 MUSIC-BARRAY-1 { ; }
15|{ : MB2 } 0 MUSIC-BARRAY-2 { ; } -->
+-----Block      71-----
0|( MUSIC PROCESSOR COMANDS ) HEX ( data,PORT )
1|{ : MASTER } 10 B, { ; } { : ATONE } 11 B, B, { ; }
2|{ : BTONE } 12 B, B, { ; } { : CTONE } 13 B, B, { ; }
3|{ : VIBS } 14 B, B, { ; } { : ABVOLS } 16 B, B, { ; }
4|{ : MCVOLS } 15 B, B, { ; } { : NOISE } 17 B, B, { ; }
5|( range,disp.,port ) { : RDRNDNTE } 0 B, B, B, B, { ; }
6|( range,port ) { : RRNDNTE } 0 B, B, 0 B, B, { ; }
7|( port ) { : RNDNTE } 0 B, B, 0 B, FF B, { ; }
8|{ : DURATION } 1 B, B, { ; } { : PLAY } 3 B, { ; }
9|( address to cont. at ) { : LDPCC } 2 B, , { ; }
10|( time,#A ) { : ANOTE } ATONE DURATION { ; }
11|( time,#B ) { : BNOTE } BTONE DURATION { ; }
12|( time,#C ) { : CNOTE } CTONE DURATION { ; }
13|( #A,#B,#C ) { : TONES } CTONE BTONE ATONE { ; }
14|( time,#A,#B,#C ) { : NOTES } TONES DURATION { ; }
15|-->
+-----Block      72-----
0|( MUSIC PROCESSOR COMANDS cont. ) HEX
1|{ : QUIET } 4 B, { ; } ( does an emusic )
2|( time,step,low,high ) { : RAMBLE } 5 B, B, B, B, B, { ; }
3|( time,step,low,high ) { : RAMP } 6 B, B, B, B, B, { ; }
4|( computer music generator, stepsize {1,0,-1}, duration ---- )
5| { : GENMUSIC } 7 B, B, B, { ; }
6|{ : RERAMBLE } 8 B, { ; } ( restart ramble )
7|{ : STOPRAMBLE } 9 B, { ; }
8|{ : COUNTLIMITS } 0A B, B, { ; }
9|( Format for following : timebase,stepsize,stopvalue ---- )
10|{ : MOVESTEP } 0B B, B, B, B, { ; }
11|( wait#oflms,ss,sv ) { : MOVELOWLIM } 0C B, B, B, B, { ; }
12|( hold#oflms,ss,sv ) { : MOVEHIGHLIM } 0D B, B, B, B, { ; }
13|( timebase ) { : MOVETB } 0E B, B, B, B, { ; }
14|( NOISE,tb,ss,sv ) { : MOVENOISE } 0F B, B, B, B, B, { ; }
15|-->

```

```

+-----Block      73-----
0|( MUSIC PROCESSOR COMANDS cont., STEREO STUFF and ABCRND ) HEX
1|( try - timebase, stepsize, leftvolume, ---- )
2|HEX      { : MOVESOUND } 18 B, B, B, B, { ; }
3|( also notice that stepvol is pos. for left-->right )
4|( for limited movement, use the following : # of limits ---- )
5|      { : COUNTPANS } 19 B, B, { ; }
6|( volume moving is ind. of stereo )
7|( ABvols,MCvols,tb,ss,ll,h1 ---- )
8|      { : MOVEVOLS } 1A B, B, B, B, B, B, MCVOLS ABVOLS { ; }
9|( special opcode to reload MC for fade out, STARTING MC ---- )
10|      { : HITMO } 1B B, B, { ; }
11|-->
12|
13|
14|
15|

```

```

+-----Block      74-----
0|( NOTE CONSTANTS ) HEX
1|FD C= #G0  EE C= #GS0  E1 C= #A0  D4 C= #AS0  C8 C= #B0
2|BD C= #C1  B2 C= #CS1  A8 C= #D1  9F C= #DS1  96 C= #E1
3|8D C= #F1  85 C= #FS1  7E C= #G1  77 C= #GS1  70 C= #A1
4|6A C= #AS1  64 C= #B1  5E C= #C2  59 C= #CS2  54 C= #D2
5|4F C= #DS2  4A C= #E2  46 C= #F2  42 C= #FS2  3E C= #G2
6|3B C= #GS2  37 C= #A2  34 C= #AS2  31 C= #B2  2E C= #C3
7|2C C= #CS3  29 C= #D3  27 C= #DS3  25 C= #E3  22 C= #F3
8|20 C= #FS3  1F C= #G3  1D C= #GS3  1B C= #A3  1A C= #AS3
9|18 C= #B3  17 C= #C4  15 C= #CS4  14 C= #D4  13 C= #DS4
10|12 C= #E4  11 C= #F4  10 C= #FS4  0F C= #G4  0E C= #GS4
11|0D C= #A4  0B C= #C5  0A C= #CS5  09 C= #DS5  08 C= #F5
12|07 C= #G5  06 C= #A5  05 C= #C6  04 C= #DS6  03 C= #G6
13|02 C= #C7  01 C= #G7  00 C= #G8 CC? IFTRUE ONSCR> IFEND
14|BTABLE NOTABLE 23 B, 22 B, 20 B, 1E B, 1C B, 1A B, 18 B, 17 B,
15| 16 B, 15 B, 14 B, 13 B, 12 B, 11 B, 0D B, 0B B, -->

```

```

+-----Block      75-----
0|( SIN BTABLE FOR LEFT-RIGHT PAN VOLTAGES ) DECIMAL
1|( : ^ ) ( STORE BYTES ON STACK IN RAM AS PATTERN )
2| ( -1 >R BEGIN DUP -1 = IF DROP 1 ELSE >R 0 THEN END
3| BEGIN R> DUP -1 = IF DROP 1 ELSE } B, { 0 THEN END ; )
4| -1 CONSTANT ~ ( MARK START OF PATTERN )
5|BTABLE sin-table
6|( 00-10 ) ~ 255 255 255 255 254 253 252 251 250 248 247 ^
7|( 11-21 ) ~ 245 243 241 239 237 234 231 229 226 220 220 ^
8|( 22-32 ) ~ 213 213 209 206 202 198 194 190 185 181 177 ^
9|( 33-43 ) ~ 172 167 162 157 152 147 142 137 132 126 121 ^
10|( 44-54 ) ~ 115 100 104 98 92 86 80 74 68 62 56 ^
11|( 55-63 ) ~ 50 44 38 31 25 19 13 6 0 ^
12|SUBR sin ( pass <, 0<= < <= 63, in E ) 0 D MVI,
13| 0 sin-table R LXI, D DAD, M A MOV, RET,
14|
15|-->

```



```

+-----Block      76-----
0|( HELPING SUBR's for MUSCPU *** NOTICE *** ) HEX
1|( The MUSPC rides in HL for the course of the MUSCPU )
2|( EACH MUSCPU LOADS ITS STARTING RAM IN IY )
3|SUBR PCJUMP ( reload MUSPC )
4| M E MOV, H INX, M D MOV, XCHG, ( leave in HL )
5| L MUSPC Y STX, H MUSPC 1+ Y STX, ( store ) RET,
6|SUBR portout ( pass value in A, port in C )
7| A E MOV, 17 A MVI, C CMP, ( all ports are 10-17 )
8| 0>=, IF, ( check for bad values )
9| 8 SUI, ( bottom ) C CMP, 0<, IF, ( oked )
10| 18 A MVI, C SUB, SOUNDBOX Y SUBX, NEG, A C MOV, E OUTP,
11| THEN, THEN, A XRA, RET,
12|SUBR babs ( byte absolute value )
13| 7 A BIT, 0<>, IF, NEG, 7 A RES, THEN, RET,
14|CODE BABS H POP, L A MOV, babs CALL, A L MOV, H PUSH, NEXT
15|-->

+-----Block      77-----
0|( HELPING SUBR's cont. ) HEX
1|SUBR LIMITCOUNT ( detect Music-State-transition if completed )
2| LIMCOUNTER Y A LDX, A ORA, 0<>, IF,
3| A DCR, A LIMCOUNTER Y STX, 0=, IF, ( done )
4| A RAMBLEFLAG Y STX, ( stop ramble ) 1 MST Y MVIX,
5| THEN, THEN, RET,
6|SUBR PANOUTS ( pass location in E )
7| PANPORT# Y C LDX, E B MOV, ( save ) sin CALL,
8| A OUTP, 3F A MVI, ( 64 steps ) B SUB, A E MOV,
9| sin CALL, ( enter table from bottom ) C INR, A OUTP,
10| RET,
11|-->
12|
13|
14|
15|

+-----Block      78-----
0|( MUSIC PROCESSOR- emusic )
1|DATA ENDMUS ASM PLAY
2|
3|SUBR emusic ( ** each EMUSIC passes vector addr in DE' )
4| MUSPC H LXI, D DAD, ENDMUS B LXI, C M MOV, H INX, B M MOV,
5| A XRA, 6 H LXI, D DAD, ( skip MUSPC,STARTPC,SOUNDBOX,PANPORT# )
6| ENDMUSRAM BEGMUSRAM - 6 - B MVI,
7| BEGIN, A M MOV, H INX, B DCR, 0=, END,
8| SOUNDBOX H LXI, D DAD, M C MOV, 8 B MVI,
9| BEGIN, C DCR, 0 OUTP, B DCR, 0=, END,
10| EXX, RET,
11|-->
12|
13|
14|
15|

```

```

+-----Block      79-----
0|( OPCODE SUBR's, 0-4 )
1|SUBR RANDOMNOTES H PUSH, ( save PC from RND )
2| 0 D MVI, M E MOV, D PUSH, H INX, M E MOV, D PUSH, H INX,
3| M E MOV, random CALL, D POP, ( disp. ) D DAD, ( returns in HL )
4| B POP, L A MOV, portout CALL, H POP, 3 D LXI, D DAD, ( MUSPC )
5| A XRA, RET,
6|SUBR LOADTIMER M A MOV, A NOTETIMER Y STX, H INX, A XRA,
7| A COMPDURATION Y STX, A INR, RET,
8|SUBR CONTJUMP M E MOV, H INX, M D MOV, XCHG, A XRA, RET,
9|SUBR QUITJUMP ( H DCX, 3 in A ) RET,
10|SUBR QUIETYET? ( QUIET ) MULTIPLE Y DCRX,
11| 0<>, IF, STARTPC Y L LDX, STARTPC 1+ Y H LDX, A XRA,
12| ELSE, Y PUSHX, EXX, D POP, emusic CALL, 1 ORI, THEN, RET,
13|-->
14|
15|

```

```

+-----Block      80-----
0|( OPCODE SUBR's, 5-6, HL= MUSPC )
1|FORWARD RAMBLESTORES
2|SUBR RAMBLIN' A XRA, ( turn off ramp flag )
3| LABEL RAMBLESTORES A RAMPFLAG Y STX,
4| M A MOV, H INX, A HIGHLIM Y STX,
5| M A MOV, H INX, A LOWLIM Y STX,
6| M A MOV, H INX, A STEP Y STX,
7| M A MOV, H INX, A RAMBLETIMER Y STX,
8| A TIMEBASE Y STX, 1 A MVI, A RAMBLEFLAG Y STX, A DCR, RET,
9|SUBR RAMPIN' 1 A MVI, RAMBLESTORES JMP,
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      81-----
0|( OPCODE SUBR's, 8-0B, 10H )
1|SUBR MASTART ( MASTER, 10H ) SOUNDBOX Y A LDX, 8 SUI, A C MOV,
2| M A MOV, H INX, A MOVALUE Y STX,
3| A OUTP, A XRA, RET,
4|SUBR RAMBLE-ON 1 A MVI, A RAMBLEFLAG Y STX, A XRA, RET,
5|SUBR RAMBLE-OFF A XRA, A RAMBLEFLAG Y STX, RET,
6|SUBR LIMITRAMBLE ( set up LIMCOUNTER )
7| 1 A MVI, A RAMBLEFLAG Y STX, M A MOV, H INX,
8| A LIMCOUNTER Y STX, A XRA, RET,
9|SUBR STEPMOVIN' M A MOV, H INX, A STOPSTEPS Y STX,
10| M A MOV, H INX, A DICOFASTER Y STX, M A MOV, H INX,
11| A STEPTIMEBASE Y STX, A STEPTIMER Y STX, A XRA, RET,
12|-->
13|
14|
15|

```

```

+-----Block      82-----
0|( OPCODES 0C-0F )
1|SUBR LOWMOVIN' M A MOV, H INX, A STOPLOWLIM Y STX,
2| M A MOV, H INX, A LOWSTEP Y STX, M A MOV, H INX, A LOW# Y STX,
3| A LOWCOUNTER Y STX, A XRA, RET,
4|SUBR HIGHMOVIN' M A MOV, H INX, A STOPHIGHLIM Y STX,
5| M A MOV, H INX, A HIGHSTEP Y STX, M A MOV, H INX,
6| A HIGH# Y STX, A HIGHCOUNTER Y STX, A XRA, RET,
7|SUBR TBMOVIN' M A MOV, H INX, A STOPTB Y STX, M A MOV, H INX,
8| A TBSTEP Y STX, M A MOV, H INX, A TBTB Y STX, A TBTIMER Y STX,
9| A XRA, RET,
10|SUBR NOMOVIN' M A MOV, H INX, A NOSTOP Y STX, M A MOV, H INX,
11| A NOSTEP Y STX, M A MOV, H INX, A NOTIMER Y STX,
12| A NOTIMEBASE Y STX, SOUNDBOX Y C LDX, C DCR,
13| M A MOV, H INX, A NOVALUE Y STX, A OUTP, A XRA, RET,
14|-->
15|

+-----Block      83-----
0|( OPCODES 11-16 I/O PORT OUTPUTS and PAN COUNTING, 1AH ) HEX
1|SUBR OPPORT ( 11H-14H, 16-17H )
2| RRC, A C MOV, M A MOV, H INX, portout JMP,
3|SUBR MCMOVIN' ( 15H )
4| RRC, A C MOV, M A MOV, H INX, A MCTRACKER Y STX, portout JMP,
5|SUBR NOISEPORT ( 17H )
6| RRC, A C MOV, M A MOV, H INX, A NOVALUE Y STX, portout JMP,
7|SUBR SOUNDMOVIN' ( 18H ) M E MOV, H INX, E LEFTPAN Y STX,
8| H PUSH, PANOUTS CALL, H POP, ( init )
9| M A MOV, H INX, A PANSTEP Y STX,
10| M A MOV, H INX, A PANTIMEBASE Y STX,
11| A PANTIMER Y STX, FF PANCOUNTER Y MVIX, A XRA, RET,
12|SUBR PANLIMITCOUNTIN' ( 19 )
13| M A MOV, H INX, A PANCOUNTER Y STX,
14| PANTIMEBASE Y A LDX, A PANTIMER Y STX, A XRA, RET,
15|-->

+-----Block      84-----
0|( STEREO OPCODE 1A, THUMPER 1B, MUSIC GENERATOR 07H ) HEX
1|SUBR VOLMOVIN' ( 1AH )
2| M A MOV, H INX, A VOLHIGHLIM Y STX,
3| M A MOV, H INX, A VOLLOWLIM Y STX,
4| M A MOV, H INX, A VOLSTEP Y STX,
5| M A MOV, H INX, A VOLTIMEBASE Y STX,
6| 1 VOLTIMER Y MVIX, A XRA, RET,
7|SUBR MOHITTIN' ( 1B )
8| 1 SYNCMO Y MVIX, ( turn on THUMPER-sync ) 3 A MVI, THUMPCOUNTER
9| STA, M A MOV, H INX, A STARTMC Y STX, A XRA, RET,
10|SUBR MUSICIN' ( 07 )
11| M A MOV, H INX, A COMPDURATION Y STX, 1 NOTETIMER Y MVIX,
12| M A MOV, H INX, A COMPTB Y STX, COMPTB COMPTIMER Y MVIX,
13| 4 ATRACKER Y MVIX, 8 BTRACKER Y MVIX, 0E CTRACKER Y MVIX,
14| 8 MOTRACKER Y MVIX, A XRA, RET, ( zero )
15|-->

```

```

+-----Block      85-----
0|( OPCODE ADDRESS TABLE and FORWARDS ) HEX
1|      1B C= #-OF-OPCODES
2|F= process F= endprocess F= MUSEND
3|TABLE OPADDRESSES
4|      RANDOMNOTES , LOADTIMER , CONTJUMP , QUITJUMP ,
5|      QUITYET? , RAMBLIN' , RAMPIN' , MUSICIN' ,
6|      RAMBLE-ON , RAMBLE-OFF , LIMITRAMBLE , STEPMOVIN' ,
7|      LOWMOVIN' , HIGHMOVIN' , TBMOVIN' , NOMOVIN' ,
8|      MASTART , 3 0 << OPPORT , >>
9|      OPPORT , MCMOVIN' , OPPORT , NOISEPORT ,
10|     SOUNDMOVIN' , PANLIMITCOUNTIN' , VOLMOVIN' , MOHITTIN' ,
11|-->
12|
13|
14|
15|

```

```

+-----Block      86-----
0|( COMPMUSIC's +-disp., 15MOD, NOTABLE and THUMPLOCATION ) HEX
1|BTABLE THUMPLOCATION ( where to locate sound in stereo image )
2| 3F B, 2A B, 15 B, 00 B,
3|BTABLE NOTABLE ASM ( 3 octave range )
4| #G0 B, #A0 B, #B0 B, #C1 B, #D1 B, #E1 B, #FS1 B,
5| #G1 B, #A1 B, #B1 B, #C2 B, #D2 B, #E2 B, #FS2 B,
6| #G2 B, #A2 B, #B2 B, #C3 B, #D3 B, #E3 B, #FS3 B,
7|SUBR +-disp. ( change A to a + or - 3-bit # )
8| 0 A BIT, 0<>, IF, ( neg ) F8 ORI, ELSE, 7 ANI, THEN, RET,
9|SUBR 15MOD ( base 21decimal ) 15 CPI, 0>=, IF, 7 SUI,
10| ( only adjust note down 1 octave ) THEN, A ORA, 0<, IF,
11| ( adjust up 1 octave ) 7 ADI, THEN, RET,
12|SUBR UP-AN-OUT ( pass index in A ) EXX,
13| 0 NOTABLE D LXI, 0 H MVI, A L MOV,
14| D DAD, C INR, M A MOV, A OUTP, ( outp note ) EXX, RET,
15|-->

```

```

+-----Block      87-----
0|( STEREO STUFF, LIMITCOUNTING )
1|( PANLIMITS- achieving limits of volumes per channel )
2|SUBR PANLIMIT ( A&E= LEFTVOL, D=tb, H=counter, L=stepvol )
3|  H INR, 0<>, IF, ( wasn't FF ) H DCR, H DCR, ( counted )
4|  0=, IF, ( counted down )
5|  1 MST Y MVIX, ( detect state transition ) 0 D MVI,
6|  THEN, ELSE, H DCR, ( back to FF )
7|  THEN, A XRA, L SUB, ( change step sign ) A L MOV,
8|  RET, ( zero or non-zero )
9|
10|
11|-->
12|
13|
14|
15|

```

```

+-----Block      88-----
0|( ** MUSCPU **                      **STEREO** ) HEX
1|SUBR MUSCPU ( runs in interrupt )
2| A XRA, MST Y CMPX, 0<>, IF, RET, THEN, ( no background )
3| NOTETIMER Y CMPX, 0<>, IF, NOTETIMER Y DCRX, 0=, IF,
4| COMPDURATION Y CMPX, 0<>, IF, 2 A MVI, NOTECOUNTER Y DCRX,
5| ELSE, A INR, THEN,
6| A MST Y STX, THEN, THEN, ( * PANNER * )
7|-->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      89-----
0|( ** MUSCPU **                      **STEREO** ) HEX
1| LEFTPAN Y E LDX, ( setups for PANNER )
2| PANTIMER Y CMPX, ( all vectoring routines follow )
3| 0<>, IF, ( timer on ) PANTIMER Y DCRX, 0=, IF, ( expired )
4| PANTIMEBASE Y D LDX, PANCOUNTER Y H LDX, PANSTEP Y L LDX,
5|( A&E=leftvol, D=timer-reload, value H=counter, L=stepvol )
6| E A MOV, L ADD, ( newpan ) A E MOV, A ORA,
7| 0<, IF, ( low limit ) 0 E MVI, PANLIMIT CALL,
8|( ~ screen 220, PANLIMIT zaps D, so a zero timer comes in )
9| ELSE, 3F CPI, >=, IF, ( 64 ) 3F E MVI, PANLIMIT CALL,
10| THEN, THEN, L PANSTEP Y STX, D PANTIMER Y STX,
11| E LEFTPAN Y STX, ( PANOUTS creams E ) H PANCOUNTER Y STX,
12| THEN, THEN, PANOUTS CALL, ( pass location in E )
13|-->
14|
15|

```

```

+-----Block      90-----
0|( MUSCPU                      NOTETIMER,MOSYNC )
1|A XRA, RAMBLEFLAG Y CMPX, ( * RAMBLER * )
2| 0<>, IF, RAMBLETIMER Y DCRX, 0=, IF,
3| STEP Y L LDX, ( setup ) SOUNDBOX Y A LDX, 8 SUI, A C MOV,
4| MOVALUE Y A LDX, L ADD, ( step ) A MOVALUE Y STX, ( upit )
5| A OUTP, A D MOV, A XRA, SYNCMO Y CMPX, 0<>, IF, EXX,
6|( * special opcode for thumping sound * ) STARTMC Y A LDX,
7| A MCTRACKER Y STX, TIMEBASE Y A LDX, ( keep durations close )
8| 0F0 ANI, RRC, RRC, RRC, RRC, A INR, A VOLTMEBASE Y STX,
9| THUMPCOUNTER H LXL, Y DCR, M C MOV, 0=, IF, A M MVI, THEN,
10| 0 B MVI, 0 THUMPLOCATION H LXL, R DAD, ( stereo loc. )
11| M E MOV, E LEFTPAN Y STX, ( let PANNER refresh ) PANOUTS CALL,
12| EXX, THEN,
13|-->
14|
15|

```

```

+-----Block      91-----
0| ( MUSCPU cont.,          MORAMBLE, LOWMOVER, HIGHMOVER )
1| ( HIGHMOVER )
2| LOWLIM Y A LDX, A DCR, D CMP, <, IF,
3| HIGH# Y A LDX, A ORA, 0<>, IF,
4| HIGHCOUNTER Y DCRX, 0=, IF, A C MOV,
5| HIGHLIM Y A LDX, HIGHSTEP Y ADDX,
6| STOPHIGHLIM Y CMPX, =, IF, 0 HIGH# Y MVIX, THEN,
7| A HIGHLIM Y STX, C HIGHCOUNTER Y STX, THEN, THEN,
8| LIMITCOUNT CALL, RAMPFLAG Y A LDX, 3 CPI, 0=, IF,
9| ( just came from LOWMOVER ) 1 A MVI, A RAMPFLAG Y STX, ELSE,
10| A ORA, 0<>, IF, ( RAMP )
11| RAMPFLAG Y INRX, ( tell LOWMOVER donothing )
12| HIGHLIM Y A LDX, L SUB, A MOVALUE Y STX, ELSE, ( change sign )
13| L SUB, A STEP Y STX, THEN, THEN, THEN,
14| -->
15|

+-----Block      92-----
0| ( MUSCPU cont.,          MORAMBLE cont., STEPMOVER )
1| HIGHLIM Y A LDX, D CMP, >=, IF, ( at limit )
2| ( LOWMOVER )
3| LOW# Y A LDX, A ORA, 0<>, IF, ( not dead already )
4| LOWCOUNTER Y DCRX, 0=, IF, A C MOV,
5| LOWLIM Y A LDX, LOWSTEP Y ADDX,
6| STOPLOWLIM Y CMPX, =, IF, 0 LOW# Y MVIX, THEN,
7| A LOWLIM Y STX, C LOWCOUNTER Y STX, THEN, THEN,
8| LIMITCOUNT CALL, RAMPFLAG Y A LDX,
9| 2 CPI, 0=, IF, RAMPFLAG Y DCRX, ELSE,
10| A ORA, 0<>, IF, ( ramp ) 3 A MVI, A RAMPFLAG Y STX,
11| ( tell HIGHMOVER to donutting )
12| LOWLIM Y A LDX, L SUB, A MOVALUE Y STX, ELSE, L SUB,
13| A STEP Y STX, THEN, THEN, THEN,
14| TIMEBASE Y A LDX, A RAMBLETIMER Y STX, THEN, THEN,
15| -->

+-----Block      93-----
0| ( MUSCPU          VOLUME MOVING )
1| VOLTIMER Y A LDX, A ORA, 0<>, IF, A DCR, 0=, IF,
2| ( D=MCtracker, E=M0set, H=limitcheck, L=stepsize )
3| MCTRACKER Y D LDX, D A MOV, 0F0 ANI, A E MOV, D A MOV,
4| E SUB, VOLSTEP Y L LDX, L ADD, ( update tracker ) A D MOV,
5| VOLHIGHLIM Y H LDX, H CMP, 0>=, IF, H D MOV, ( check top )
6| A XRA, L SUB, A L MOV, ( change dir. )
7| ELSE, A DCR, VOLLOWLIM Y H LDX, H CMP, 0<, IF,
8| H D MOV, ( low end ) A XRA, L SUB, A L MOV, ( change )
9| THEN, THEN,
10| L VOLSTEP Y STX, D A MOV, ( Cvals ) RRC, RRC, RRC, RRC,
11| D ADD, ( ABvals is CCvals ) SOUNDBOX Y C LDX, C DCR, C DCR,
12| A OUTP, C DCR, D A MOV, E ORA, ( mosetting )
13| A MCTRACKER Y STX, A OUTP,
14| VOLTIMBASE Y A LDX, THEN, A VOLTIMER Y STX, THEN,
15| -->

```

```

+-----Block      94-----
0|( MUSCPU                      STEPMOVER, COMPDURMOVER )
1| STEPTIMER Y A LDX, A ORA, 0<>, IF, ( work on it )
2| STEPTIMER Y DCRX, 0=, IF,
3| STEP Y A LDX, A E MOV, ( save for sign ) babs CALL, ( abs )
4| BIGOFASTEP Y ADDX, ( positive value )
5| STOPSTEPS Y CMPX, ( positive ) 0<>, IF, ( not done )
6| STEPTIMEBASE Y A LDX, A STEPTIMER Y STX, THEN, 7 E BIT,
7| 0<>, IF, NEG, THEN, A STEP Y STX, ( store ) THEN, THEN,
8|( COMP-DURATION MOVER ) COMPDURATION Y A LDX, A ORA, 0<>, IF,
9| COMPTIMER Y DCRX, 0=, IF, COMPTB COMPTIMER Y MVIX, ( equate )
10| COMPSTEP Y ADDX, A DCR, 0<, IF, ( 1=>CARRY, 81=>NC )
11| 0 COMPDURATION Y MVIX, ( both stop )
12| 1 MST Y MVIX, ( tell background we're there ) ELSE,
13| A INR, A COMPDURATION Y STX, THEN, THEN, THEN,
14|-->
15|

+-----Block      95-----
0|( MUSCPU cont.,                      TBMOVER, NOMOVER )
1|( TBMOVER timebase ) TBSTEP Y A LDX, A ORA, 0<>, IF,
2| ( not done ) TBTIMER Y DCRX, 0=, IF,
3| TIMEBASE Y ADDX, A TIMEBASE Y STX, STOPTB Y SUBX,
4| 0=, IF, A TBSTEP Y STX, THEN,
5| TBTB Y A LDX, A TBTIMER Y STX, THEN, THEN,
6|( NOMOVER noise mover ) NOTIMER Y A LDX, A ORA, 0<>, IF,
7| NOTIMER Y DCRX, 0=, IF, ( update )
8| SOUNDBOX Y C LDX, C DCR, ( output port )
9| NOVALUE Y A LDX, NOSTEP Y ADDX, A NOVALUE Y STX, A OUTP,
10| NOSTOP Y CMPX, ( timer to stop? )
11| 0<>, IF, ( not done, reload )
12| NOTIMEBASE Y A LDX, A NOTIMER Y STX, THEN, THEN, THEN,
13| RET,
14|-->
15|

+-----Block      96-----
0|( ** MUSIC INTERRUPTER **          COMPUTER MUSIC ) HEX
1| SUBR MUSINTERP ( music interrupter, pre-load IY )
2| ASSEMBLE MST Y A LDX, A ORA, ( check overrun flag )
3| 0=, IF, RET, THEN, A DCR, 0=, IF, ( AHAI! state transition )
4| process JMP, THEN, ( AHHA HA!! MST=2 => COMPUTER MUSIC )
5| COMPDURATION Y A LDX, A NOTETIMER Y STX, ( compmusic duration )
6| random CALL, ( use 3 reg. melody trackers, 1 for MO tracker )
7| NOTECOUNTER Y A LDX, A ORA, 0<, IF, ( new key ) EXX,
8| LDAR, ( gen new wait # ) 1 E MVI, 7 D MVI, ( # = 2^n )
9| BEGIN, RAR, CY, IF, E SLAR, THEN, D DCR, 0=, END, E SLAR,
10| ( 2^2 to 2^8 ) E NOTECOUNTER Y STX, EXX, 1 L MVI, ( inc MO )
11| ELSE, 0 L MVI, ( leave MO ) THEN,
12| L A MOV, ( do anyhow for setup )
13| MOTRACKER Y ADDX, 0F ANI, ( 0-15 ) A MOTRACKER Y STX,
14|-->
15|

```

```

+-----Block      97-----
0|( MUSCPU cont.,          RANDOM NOTES )
1| EXX, 0 MOTABLE H LXI, A E MOV, 0 D MVI, D DAD, M A MOV,
2| A MOVALUE Y STX, A B MOV, ( save from soundbox )
3| SOUNDBOX Y A LDX, 8 SUI, A C MOV, 20 CPI, <, IF, ( low chip )
4| B OUTP, C D MOV, 40 ADI, A C MOV, B OUTP, ( MO ) D C MOV, THEN,
5| EXX, H A MOV, ( use this # for disp. to index ) +-disp. CALL,
6| ( +7 to -7 ) ATRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
7| A ATRACKER Y STX, UP-AN-OUT CALL,
8| E A MOV, ( next note ) +-disp. CALL, ( +7 to -6 disp. )
9| BTRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
10| A BTRACKER Y STX, UP-AN-OUT CALL, ( get from NOTE TABLE )
11| D A MOV, ( next disp. ) +-disp. CALL, ( +7 to -6 )
12| CTRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
13| A CTRACKER Y STX, UP-AN-OUT CALL,
14| ( done with notes ) MUSEND JMP,
15|-->

+-----Block      98-----
0|( MUSCPU cont.,          PROCESS the score, )
1| LABEL process MUSPC Y L LDX, MUSPC 1+ Y H LDX,
2| BEGIN, ( MUSPC in HL until done )
3| M A MOV, H INX, #-0F-OPCODES 1+ CPI, ( bad opcode check )
4| <, IF, EXX, ( swap to keep MUSPC around )
5| endprocess H LXI, H PUSH, ( ret to end of process )
6| 0 OPADDRESSES H LXI, ( get address of opcode verb )
7| RLC, ( words ) A E MOV, 0 D MVI, D DAD,
8| M E MOV, H INX, M D MOV, D PUSH, ( RET to routine )
9| EXX, ( put MUSPC in HL ) RET,
10| ELSE, 1 ORI, ( quit ) THEN,
11| LABEL endprocess A ORA, 0<>, END, ( opverbs return non-0 or 0 )
12| L MUSPC Y STX, H MUSPC 1+ Y STX,
13| LABEL MUSEND 0 MST Y MVIX, ( let interrupts run ) RET,
14| ASSEMBLE>
15|-->

+-----Block      99-----
0|( MUSIC PROCESSOR-      MUSCPUS PUT TOGETHER ) HEX
1| SUBR MUSCPUS
2| MUSICFLAG LDA, A ORA, 0<>, IF, Y PUSHX, ( F4 A MVI, 0 OUT, )
3| 0 MUSIC-BARRAY-1 Y LXIX, MUSCPU CALL, ( A2 A MVI, 0 OUT, )
4| 0 MUSIC-BARRAY-2 Y LXIX, MUSCPU CALL, ( 53 A MVI, 0 OUT, )
5| Y POPX, THEN, RET,
6| SUBR busaround ( back-music-ground )
7| MUSICFLAG LDA, A ORA, 0<>, IF, Y PUSHX, ( F4 A MVI, 4 OUT, )
8| 0 MUSIC-BARRAY-1 Y LXIX, MUSINTERP CALL, ( A2 A MVI, 4 OUT, )
9| 0 MUSIC-BARRAY-2 Y LXIX, MUSINTERP CALL, ( 53 A MVI, 4 OUT, )
10| Y POPX, THEN, RET,
11| CODE BMS ( code level back-music-ground ) B PUSH,
12| busaround CALL, B POP, NEXT
13|
14|-->
15|

```



```

+-----Block      100-----
0|( MUSIC PROCESSOR- ALL xmusics NEED AN IY LOAD ) HEX
1|
2|SUBR loadpc L MUSPC Y STX, L STARTPC Y STX,
3| H MUSPC 1+ Y STX, H STARTPC 1+ Y STX, RET,
4|-->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      101-----
0|( MUSCPU SUBROUTINE CALLS )
1|( * SET SCORE IN HL, RAM IN IY, MULTIPLE IN E if req. * )
2|SUBR bmusic
3| PRIORITY Y A LDX, A ORA, 0=, IF, 1 MST Y MVIX,
4| A NOTETIMER Y STX, A INR, A MULTIPLE Y STX, loadpc JMP,
5| ( leave MST=1 for BMS ) THEN, RET,
6|SUBR pmusic 1 MST Y MVIX, Y PUSHX, EXX, D POP, emusic CALL,
7| 1 A MVI, A MST Y STX, A MULTIPLE Y STX, A PRIORITY Y STX,
8| loadpc JMP,
9|SUBR mmusic PRIORITY Y A LDX, A ORA, 0=, IF, 1 MST Y MVIX,
10| A NOTETIMER Y STX, E MULTIPLE Y STX, loadpc JMP, THEN, RET,
11|SUBR mpmusic 1 MST Y MVIX, Y PUSHX, EXX, D POP, emusic CALL,
12| 1 A MVI, A MST Y STX, A PRIORITY Y STX, E MULTIPLE Y STX,
13| loadpc JMP,
14|-->
15|
+-----Block      102-----
0|( MUSIC PROCESSOR- EMUSIC, BMUSIC, ... )
1|CODE EMUSIC EXX, 0 MUSIC-BARRAY-1 D LXI, ( pass to emusic )
2| MST H LXI, D DAD, 1 M MVI, ( make non-zero )
3| SOUNDBOX H LXI, D DAD, CHIP1 M MVI,
4| PANPORT# H LXI, D DAD, PANPORT1 M MVI,
5| emusic CALL, ( musicoverrun flag is zeroed last ) NEXT
6|( *** ALWAYS CALL EMUSIC AS AN INIT IN PROGRAM *** )
7|CODE BMUSIC H POP, Y PUSHX,
8| 0 MUSIC-BARRAY-1 Y LXI, bmusic CALL, Y POPX, NEXT
9|CODE PMUSIC H POP, Y PUSHX,
10| 0 MUSIC-BARRAY-1 Y LXI, pmusic CALL, Y POPX, NEXT
11|CODE MMUSIC H POP, D POP, Y PUSHX,
12| 0 MUSIC-BARRAY-1 Y LXI, mmusic CALL, Y POPX, NEXT
13|CODE MPMUSIC H POP, D POP, Y PUSHX,
14| 0 MUSIC-BARRAY-1 Y LXI, mpmusic CALL, Y POPX, NEXT
15|-->

```

```

+-----Block      103-----
0|( MUSIC PROCESSOR- E2MUSIC, B2MUSIC, ... )
1|CODE E2MUSIC EXX, 0 MUSIC-BARRAY-2 D LXI, ( pass to emusic )
2| MST H LXI, D DAD, 1 M MVI, ( make non-zero )
3| SOUNDBOX H LXI, D DAD, CHIP2 M MVI,
4| PANPORT# H LXI, D DAD, PANPORT2 M MVI,
5| emusic CALL, ( musiccoverun flag is zeroed last ) NEXT
6|( *** ALWAYS CALL E2MUSIC AS AN INIT IN PROGRAM *** )
7|CODE B2MUSIC H POP, Y PUSHX,
8| 0 MUSIC-BARRAY-2 Y LXIX, bmusic CALL, Y POPX, NEXT
9|CODE P2MUSIC H POP, Y PUSHX,
10| 0 MUSIC-BARRAY-2 Y LXIX, pmusic CALL, Y POPX, NEXT
11|CODE M2MUSIC H POP, D POP, Y PUSHX,
12| 0 MUSIC-BARRAY-2 Y LXIX, mmusic CALL, Y POPX, NEXT
13|CODE MP2MUSIC H POP, D POP, Y PUSHX,
14| 0 MUSIC-BARRAY-2 Y LXIX, mpmusic CALL, Y POPX, NEXT
15|-->

+-----Block      104-----
0|( CLEAR ANY PRIORITY ON THE MUSIC PROCESSOR )
1|CODE UNPRIOR EXX, Y PUSHX,
2|0 MUSIC-BARRAY-1 Y LXIX, 0 PRIORITY Y MVIX,
3|0 MUSIC-BARRAY-2 Y LXIX, 0 PRIORITY Y MVIX,
4|Y POPX, EXX, NEXT
5|
6|( SHUT UP VERB TO QUIET BOTH MUSIC PROCESSORS )
7|: SHUTUP EMUSIC E2MUSIC ;
8|DECIMAL
9|-->
10|
11|
12|
13|
14|
15|

+-----Block      105-----
0|( JAYS VIDEO GAME GOODIES )
1|HEX : CL 0 4000 4000 FILL ;
2|DECIMAL { : BINARY } 2 BASE ! { ; }
3|{ : QUAD } 4 BASE ! { ; }
4|HEX
5|CC? IFTRUE : GRAPHICS MAP 0 FB OUTP ; OTHERWISE : GRAPHICS ;
6|IFEND
7|: XY 100 * SWAP 40 * SWAP ;
8|DECIMAL
9|CODE DOIT H POP, PCHL, NEXT
10|: INIT GRAPHICS 1 8 OUTP 204 10 OUTP 43 9 OUTP ;
11|-->
12|
13|
14|
15|

```

```

+-----Block      106-----
0|( QUEUE - VECTOR MANIPULATION ROUTINES )
1|( THE QUEUE IS MAINTAINED AS A DOUBLE LINKED CIRCULAR LIST )
2|CC? IFTRUE <ONSCR IFEND
3|0 C= PQS ( STATUS ) 1 C= PQFL 2 C= PQFH ( FORWARD LINK )
4|3 C= PQBL 4 C= PQBH ( BACKWARD LINK )
5|5 C= PQRL 6 C= PQRH ( ROUTINE ) 7 C= PQTb ( TIME BASE )
6|8 C= VXZW ( EXCLUSION ZONE WIDTH ) 9 C= VAUXS ( AUX STATUS )
7|10 C= VATMR ( ANIMATION TIMER )
8|11 C= VTLL 12 C= VTLH ( TIME LIMIT )
9|13 C= VXL 14 C= VXH 15 C= VDXL 16 C= VDXH ( X AND DX )
10|17 C= VDDXL 18 C= VDDXH ( DDX )
11|19 C= VYL 20 C= VYH 21 C= VDYL 22 C= VDYH ( Y AND DY )
12|23 C= VDDYL 24 C= VDDYH ( DDY )
13|25 C= VSAL 26 C= VSAH ( SCREEN ADDR ) 27 C= VMAGIC ( MAGIC )
14|28 C= VXPAND ( EXPANDER ) 29 C= VPATL 30 C= VPATH ( PAT ADDR )
15|-->

```

```

+-----Block      107-----
0|( VECTOR FIELD EQUATES CONTINUED )
1|31 C= VFVPL 32 C= VFVPH ( FORMATION VECTOR POINTERS )
2|33 C= VPCL 34 C= VPCH ( ANIMATION PROGRAM COUNTER )
3|35 C= VSPL 36 C= VSPH ( ANIMATION STACK POINTER )
4|37 C= VPTBL 38 C= VPTBH ( ANIMATION PATTERN TABLE )
5|39 C= VIRL 40 C= VIRH ( INTERCEPT CHECK ROUTINE )
6|41 C= VINTER ( INTERCEPT CODE ) 41 C= VRACK ( RACK CODE )
7|42 C= VFNLPL 43 C= VFNLPH ( FINAL ANIMATION PATTERN )
8|44 C= VSHFTA ( MAGIC REG USED IN LAST WRITE )
9|45 C= VIDENT ( IDENTITY CODE - WHAT I AM )
10|46 C= VFXBL 47 C= VFXBH ( FORMATION X BIAS )
11|48 C= VFYBL 49 C= VFYBH ( FORMATION Y BIAS )
12|50 C= VASTKS ( ANIMATION STACK AREA START )
13|-->
14|
15|

```

```

+-----Block      108-----
0|( STATUS BIT EQUATES )
1|7 C= PQSRH ( RUN/HALT )
2|6 C= PQSDW ( DONT WRITE )
3|5 C= PQSDE ( DONT ERASE )
4|4 C= PQSDF ( DONT FREE )
5|3 C= PQSDS ( DONT SCREEN SYNCHONIZE )
6|2 C= PQSUFP ( USE FINAL PATTERN ON HALT )
7|1 C= PQSNMT ( NO MASTER TIME LIMIT )
8|0 C= PQSFRZ ( OFFSTAGE FREEZE )
9|( AUXILLARY STATUS BITS )
10|7 C= ASFLOK ( FORMATION MEMBER IS LOCKED IN )
11|( EQUATES FOR VECTOR HEAD )
12|0 C= QFL 1 C= QFH 2 C= QBL 3 C= QBH
13|CC? IFTRUE <ONSCR> IFEND
14|-->
15|

```

```

+-----Block      109-----
0|( VGS                      VWRITE )
1|SUBR vwrite ( Write from a vector structure )
2| ( in- IX=vmagic of proper vector ; vector equates set )
3| ( WRTSYS 0<> for pattern board 0= for software write )
4| ( out- pattern on screen ; scradr and shift saved for VERASE )
5| VXPAND X B LDX, VMAGIC X C LDX, VXH X D LDX, VXL X E LDX,
6| VPATH X H LDX, VPATL X L LDX, H PUSH, XTIY,
7| VYH X H LDX, VYL X L LDX,
8|ffrelabs CALL, ( calculates magic add. )
9| H VSAH X STX, L VSAL X STX, ( set scradr for erase )
10| writep CALL, ( write it )
11| C VSHFTA X STX, ( save shift for erase ) Y POPX, RET,
12|-->
13|
14|
15|

```

```

+-----Block      110-----
0|( VGS                      VERASE )
1|SUBR verase ( does pattern board erase from vector IX )
2| ( in- IX=vmagic of proper vector ; vector equates set ;
3| scradr and shift saved from VWRITER )
4| ( WRTSYS 0<> for pattern board 0= for software write )
5| ( out- erased pattern from screen )
6| VXPAND X B LDX, VSHFTA X C LDX, VPATH X H LDX, VPATL X L LDX,
7| H PUSH, XTIY,
8| VSAH X H LDX, VSAL X L LDX,
9| writep CALL, Y POPX, RET,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      111-----
0|( GLOBAL GAME RAM AREA START )
1|HEX RAMBASE 300 + C= FIRSTRAMADDR FIRSTRAMADDR VPTR ! DECIMAL
2|0 V= FBCOUNTER 3 BA= P1SCR 3 BA= P2SCR 0 V= MISSION
3|0 V= P1ACT 0 V= P2ACT 0 V= MISSIONCTR
4|0 V= SKILLFACTOR 0 V= PLAYERUP
5|0 V= NPLAYERS 0 V= OTHERFBCTR 0 V= OTHERSKILLF
6|0 V= MUTHAX 0 V= MUTHAY
7|
8|4 BARRAY vqhead : NILVQ 0 0 vqhead ! 0 2 vqhead ! ;
9|
10|FIRSTRAMADDR 60 + C= 1STCLRADDR
11|LASTRAMADDR 1STCLRADDR - 1+ C= CLRSIZE
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block 112-----
0|( NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
1|( THESE PARAMETERS TUNE THE BACKGROUND TIME SLICING )
2|0 V= BGWINDOW ( # OF LINES FOR BACKGROUND TIME SLICE )
3|0 V= BGTLMT ( BACKGROUND MINIMUM SERVICE FREQUENCY )
4|0 V= BACKGROUND RUNNING 0 V= LOCKOUTCOUNTER 0 V= BGTIMER
5|0 V= LPYC 0 V= LPFLAG
6|0 V= TIMER0 0 V= TIMER1 0 V= TIMER2 0 V= TIMER3
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block 113-----
0|( STORAGE ALLOCATOR GOODIES )
1|1STCLRADDR VPTR !
2|24 C= NODECOUNT 64 C= NODESIZE NODECOUNT NODESIZE * C= POOLSIZE
3|POOLSIZE BARRAY MEMPOOL 0 MEMPOOL VARIABLE FREELIST
4|: INITFREELIST NODECOUNT 1 DO NODESIZE I * MEMPOOL
5| NODESIZE I 1 - * MEMPOOL 1+ ! LOOP ( THREAD THRU POINTERS )
6|0 NODECOUNT 1 - NODESIZE * MEMPOOL 1+ ! 0 MEMPOOL FREELIST ! ;
7|( GET A NODE FROM ASM LANGUAGE - RETURN: HL=NODE, 0 IF CAN'T )
8|SUBR getnode FREELIST LHLD, L A MOV, H ORA, ( CHECK FOR NIL )
9| 0<>, IF, H INX, M E MOV, H INX, M D MOV, H DCX, ( FREE=NXT )
10|H DCX, FREELIST SDED, THEN, RET,
11|CODE GETNODE DI, getnode CALL, H PUSH, EI, NEXT ( TERSE ENTRY )
12|( RELEASE NODE - HL=BLOCK TO FREE )
13|SUBR freenode FREELIST LDED, FREELIST SHLD, ( LINK IN AT HEAD )
14|H INX, E M MOV, H INX, D M MOV, RET,
15|CODE FREENODE H POP, freenode CALL, NEXT -->
+-----Block 114-----
0|( ADD NODE TO QUEUE ROUTINE )
1|SUBR ADDQ ( HL = NEW, IY = HEAD )
2|QFL Y E LDX, QFH Y D LDX, E A MOV, D ORA, 0<>, IF,
3|QBL Y C LDX, QBH Y B LDX, H PUSH, H INX,
4|E M MOV, H INX, D M MOV, H INX, C M MOV, H INX, B M MOV,
5|XCHG, D POP, H INX, H INX, H INX, E M MOV, H INX, D M MOV,
6|E A MOV, B INX, B STAX, D A MOV, B INX, B STAX,
7|ELSE, L E MOV, H D MOV, H INX, E M MOV, H INX, D M MOV,
8|H INX, E M MOV, H INX, D M MOV, E QBL Y STX, D QBH Y STX,
9|THEN, E QFL Y STX, D QFH Y STX, RET,
10|CODE ADDTQ DI, EXX, H POP, XTIY, ADDQ CALL, Y POPX, EXX, EI,
11|NEXT -->
12|
13|
14|
15|

```

```

+-----Block      115-----
0|( DELETE FROM QUEUE )
1|SUBR dela ( IY=HEAD, IX=NODE TO DELETE )
2|QFL Y E LDX, QFH Y D LDX, QBL Y L LDX, QBH Y H LDX, ( HEADER )
3|A XRA, D DSBC, 0=, IF, ( IF I AM THE ONLY GUY LEFT )
4|A QFL Y STX, A QFH Y STX, A QBL Y STX, A QBH Y STX, ( NIL OUT )
5|ELSE, PQFL X E LDX, PQFH X D LDX, PQBL X C LDX, PQBH X B LDX,
6|C L MOV, B H MOV, H INX, E M MOV, H INX, D M MOV, ( F[F[N]] = F )
7|PQBL H LXI, D DAD, C M MOV, H INX, B M MOV, ( B[F[N]] = B[N] )
8|X PUSHX, H POP, XCHG, H PUSH, ( DE=NODE ADDR, TOP=NODE FORM )
9|QFL Y L LDX, QFH Y H LDX, A ANA, D DSBC, H POP,
10|0=, IF, L QFL Y STX, H QFH Y STX, ( SET HEAD TO F[N] )
11|ELSE, QBL Y L LDX, QBH Y H LDX, A ANA, D DSBC, 0=, IF,
12|C QBL Y STX, B QBH Y STX, ( SET TAIL ) THEN, THEN, THEN, RET,
13|CODE DELQ EXX, H POP, XTIY, X PUSHX, H PUSH, X POPX, dela CALL,
14|X POPX, Y POPX, EXX, NEXT
15|-->

```

```

+-----Block      116-----
0|( ADVANCE TO NEXT NODE ON QUEUE )
1|SUBR nexta ( IY = HEAD )
2|QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA, RZ,
3|H INX, M E MOV, H INX, M D MOV, ( DE=FORM[FORM[HEAD]] )
4|H DCX, H DCX, L QBL Y STX, H QBH Y STX, ( BACK[HEAD]=NODE )
5|E QFL Y STX, D QFH Y STX, ( FORM[HEAD]=FORM[NODE] )
6|RET,
7|CODE NEXTQ XTIY, nexta CALL, Y POPX, H PUSH, NEXT
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      117-----
0|( INCREMENT TIME BASES - C = TIME BASE, IY = 0 HEAD )
1|SUBR INCTB
2|QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA, ( 0 FORM, NIL CHECK )
3|RZ, L E MOV, H D MOV, BEGIN, ( QUIT IF NIL, ELSE REMEM FIRST )
4|D PUSH, PQTB D LXI, D DAD, M A MOV, C ADD, A M MOV, ( UPDATE )
5|A ANA, D DSBC, D POP, ( AND RETURN PTR TO NORMAL )
6|H INX, M A MOV, H INX, M H MOV, A L MOV, ( HL=FORM[NODE] )
7|E XRA, A B MOV, H A MOV, D XRA, B ORA, 0=, ( ONCE AROUND )
8|END, RET,
9|-->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      118-----
0|( NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
1|HEX CC? IFTRUE 62 C= IPNT OTHERWISE HERE 0F + FFF0 AND DP !
2|DATA KPNT 0 , 0 , 0 , KPNT 2 + C= IPNT IFEND
3|IPNT 2 + C= BGINTVEC
4|( LIGHT PEN INTERRUPT ROUTINE )
5|SUBR LPINT PSW PUSH, VERAf IN, LPYC STA, 8 A MVI, LPFLAG STA,
6|INMOD OUT, PSW POP, EI, RET,
7|( ROUTINE TO RETURN Y ADDRESS SCREEN IS AT )
8|SUBR GETSYC A XRA, LPFLAG STA, 18 A MVI, INMOD OUT, BEGIN,
9|LPFLAG LDA, A ANA, 0<>, END, LPYC LDA, RET,
10|DECIMAL -->
11|
12|
13|
14|
15|
+-----Block      119-----
0|( RESUME BACKGROUND - END INTERRUPT )
1|DECIMAL F= ENDINT
2|SUBR RESUMEBACKGROUND <ASSEMBLE
3|DI, BGWINDOW LDA, C ADD, A C MOV,
4|112 SUI, 52 CPI, ENDINT JRC, C A MOV, INLIN OUT,
5|BGINTVEC A MVI, INFBK OUT,
6|LABEL ENDINT Y POPX, X POPX, H POP, D POP, B POP, PSW POP,
7|EXX, EXAF, H POP, D POP, B POP, 1 A MVI, BACKGROUNDRUNNING STA,
8|BGTLMT LDA, BGTIMER STA,
9|PSW POP, EI, RET, ASSEMBLE>
10|-->
11|
12|
13|
14|
15|
+-----Block      120-----
0|( TRY TO RUN SOMETHING IN FOREGROUND )
1|F= RETPT F= TRYAGAIN
2|HEX SUBR TRYFOREGROUND <ASSEMBLE
3|A XRA, BACKGROUNDRUNNING STA, EI,
4|LABEL TRYAGAIN GETSYC CALL, A C MOV, ( C=CURRENT LINE )
5|BGTIMER LDA, A ANA, RESUMEBACKGROUND JZ, ( OR TIMER COUNTDOWN )
6|0 vahead Y LXIX, QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA,
7|RESUMEBACKGROUND JZ, H PUSH, X POPX, P0TB X A LDX, A ANA,
8|RESUMEBACKGROUND JZ, PQSDS PQS X BITX, 0=, IF,
9|VYH X A LDX, C SUB, 0<, IF, CMA, A INR, THEN,
10|VXZW X CMPX, RESUMEBACKGROUND JC, THEN, DI, nexta CALL,
11|EI, TRYAGAIN H LXI, H PUSH, PQRL X L LDX, PGRH X H LDX, PCHL,
12|ASSEMBLE>
13|-->
14|
15|

```

```

+-----Block      121-----
0|( BACKGROUND END INTERRUPT )
1|HEX SUBR BGENDI PSW PUSH, B PUSH, D PUSH, H PUSH, EXX, EXAF,
2|PSW PUSH, B PUSH, D PUSH, H PUSH, X PUSHX, Y PUSHX,
3|80 A MVI, INLIN OUT, IPNT A MVI, INFBK OUT, TRYFOREGROUND JMP,
4|F= LOCKED
5|SUBR TIMINT (ASSEMBLE ( TIMER INTERRUPT ROUTINE )
6|PSW PUSH, B PUSH, D PUSH, H PUSH, EXX, EXAF, PSW PUSH, B PUSH,
7|D PUSH, H PUSH, X PUSHX, Y PUSHX,
8|MUSCPUS CALL,
9|LOCKOUTCOUNTER LDA, A ANA, LOCKED JNZ,
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      122-----
0|( BACKGROUND END INTERRUPT )
1|TIMER0 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER0 SHLD, THEN,
2|TIMER1 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER1 SHLD, THEN,
3|TIMER2 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER2 SHLD, THEN,
4|TIMER3 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER3 SHLD, THEN,
5|BGTIMER LDA, A ANA, 0<>, IF, A DCR, BGTIMER STA, THEN,
6|1 C MVI, 0 vahead Y LXIX, INCTB CALL,
7|BACKGROUND RUNNING LDA, A ANA, TRYFOREGROUND JNZ,
8|LABEL LOCKED
9|Y POPX, X POPX, H POP, D POP, B POP, PSW POP,
10|EXX, EXAF, H POP, D POP, B POP, PSW POP, EI, RET,
11|ASSEMBLE>
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      123-----
0|( INTERRUPT START ROUTINE ) HEX
1|CODE INTSTART DI, IPNT ( SWAB ) A MVI, STAI, IPNT A MVI,
2|INFBK OUT, 1 A MVI, BACKGROUND RUNNING STA, 8 A MVI, INMOD OUT,
3|80 A MVI, INLIN OUT,
4|IM2, EI, NEXT
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```



```

+-----Block      124-----
0|( ROUTINE TO DELETE VECTOR IF STATUS SO INDICATES )
1|SUBR KILLOFF PQSRH PQS X BITX, 0=, IF, DI, 0 vqhead Y LXIX,
2|delq CALL, PQSDF PQS X BITX, 0 PQS X MVIX,
3|0=, IF, X PUSHX, H POP, freenode CALL,
4|THEN, EI, THEN, RET,
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      125-----
0|( MACROS TO GENERATE ANIMATION OPCODES ) DECIMAL
1|CC? IFTRUE <ONSCR IFEND
2|{( : SETP ) 0 B, , { ; }
3|{( : SETM ) 2 B, B, { ; }
4|{( : SETR ) 4 B, , { ; }
5|{( : SWAIT ) 6 B, B, { ; }
6|{( : ACALL ) 8 B, , { ; }
7|{( : AJMP ) 10 B, , { ; }
8|{( : SETDC ) 12 B, SWAP , , { ; }
9|{( : SETDDC ) 14 B, SWAP , , { ; }
10|{( : ARET ) 16 B, { ; }
11|{( : AHALT ) 18 B, { ; }
12|{( : SETI ) 20 B, , { ; }
13|{( : SETXC ) 22 B, , { ; }
14|{( : SETYC ) 24 B, , { ; }
15|{( : DISPL ) 26 B, SWAP B, B, { ; } -->

```

```

+-----Block      126-----
0|( MORE ANIMATION MACRO STUFF )
1|{( : AREPEAT ) 28 B, B, HERE { ; }
2|{( : ALOOP ) 30 B, , { ; }
3|{( : SETS ) 32 B, B, B, { ; }
4|{( : PATI ) 34 B, B, { ; }
5|{( : ASMCALL ) 36 B, , { ; }
6|{( : SETPT ) 38 B, , { ; }
7|{( : SETFP ) 40 B, , { ; }
8|{( : SETXZW ) 42 B, B, { ; }
9|{( : RANDOMDO ) 44 B, SWAP B, , { ; }
10|{( : SETXB ) 46 B, , { ; }
11|{( : SETYB ) 48 B, , { ; }
12|{( : SETXP ) 50 B, B, { ; }
13|{( : FOREVER ) HERE { ; }
14|{( : EVERFOR ) 52 B, , { ; }
15|CC? IFTRUE ONCCR> IFEND -->

```

```

+-----Block      127-----
0|( ANIMATION INTERPRETER ROUTINES )
1|SUBR RASETP M E MOV, H INX, M D MOV, H INX, E VPATL X STX,
2|D VPATH X STX, RET,
3|SUBR RASETM M A MOV, H INX, A VMAGIC X STX, RET,
4|SUBR RASETR M E MOV, H INX, M D MOV, H INX, E PQRL X STX,
5|D PQRH X STX, RET,
6|SUBR RAWAIT M A MOV, H INX, A VATMR X STX, L VPCL X STX,
7|H VPCH X STX, H POP, RET,
8|SUBR RACALL M C MOV, H INX, M B MOV, H INX, XCHG,
9|VSPL X L LDX, VSPH X H LDX, E M MOV, H INX, D M MOV, H INX,
10|L VSPL X STX, H VSPH X STX, C L MOV, B H MOV, RET,
11|SUBR RARET VSPL X L LDX, VSPH X H LDX, H DCX, M D MOV, H DCX,
12|M E MOV, L VSPL X STX, H VSPH X STX, XCHG, RET,
13|SUBR RAHALT PQSRH PQS X RESX, H POP, RET,
14|SUBR RASEXP M A MOV, H INX, A VXPAND X STX, RET,
15|-->

```

```

+-----Block      128-----
0|( MORE ANIMATION INTERPRETER ROUTINES )
1|SUBR RASETI M E MOV, H INX, M D MOV, H INX, E VIRL X STX,
2|D VIRH X STX, RET,
3|SUBR RASETXC M E MOV, H INX, M D MOV, H INX, E VXL X STX,
4|D VXH X STX, RET,
5|SUBR RASETYC M E MOV, H INX, M D MOV, H INX, E VYL X STX,
6|D VYH X STX, RET,
7|SUBR RAJMP M E MOV, H INX, M D MOV, XCHG, RET,
8|SUBR RASETDC M E MOV, H INX, M D MOV, H INX,
9|E VDXL X STX, D VDXH X STX,
10|M E MOV, H INX, M D MOV, H INX,
11|E VDYL X STX, D VDYH X STX, RET,
12|SUBR RASETDDC M E MOV, H INX, M D MOV, H INX, E VDDXL X STX,
13|D VDDXH X STX, M E MOV, H INX, M D MOV, H INX, E VDDYL X STX,
14|D VDDYH X STX, RET,
15|-->

```

```

+-----Block      129-----
0|( YET MORE ANIMATION INTERPRETER ROUTINES )
1|SUBR RASETREP M A MOV, H INX, VSPL X E LDX, VSPH X D LDX,
2|D STAX, D INX, E VSPL X STX, D VSPH X STX, RET,
3|SUBR RALOOP M E MOV, H INX, M D MOV, H INX, VSPL X C LDX,
4|VSPH X B LDX, B DCX, B LDAX, A DCR, 0<>, IF, B STAX, XCHG,
5|ELSE, C VSPL X STX, B VSPH X STX, THEN, RET,
6|SUBR RASETS M C MOV, H INX, M B MOV, H INX, PQS X A LDX,
7|C XRA, B ANA, C XRA, A PQS X STX, RET,
8|HEX
9|SUBR RADISP M A MOV, H INX, XCHG, RRC, RRC, A B MOV, C0 ANI,
10|A C MOV, B A MOV, S A BIT, 0<>, IF, C0 ORI, ELSE, SF ANI,
11|THEN, A B MOV, VXL X L LDX, VXH X H LDX, B DAD, L VXL X STX,
12|H VXH X STX, D LDAX, D INX, A B MOV, 0 C MVI,
13|VYL X L LDX, VYH X H LDX, B DAD, L VYL X STX, H VYH X STX,
14|XCHG, RET,
15|DECIMAL -->

```

```

+-----Block      130-----
0|( THE ABSOLUTELY LAST SCREEN OF ANIMATION INTERPRETER STUFF )
1|SUBR RAPATI M C MOV, H INX, 0 B MVI, XCHG,
2|VPTBL X L LDX, VPTBH X H LDX, B DAD, M C MOV, H INX, M B MOV,
3|C VPATL X STX, B VPATH X STX, XCHG, RET,
4|SUBR RASMCALL M E MOV, H INX, M D MOV, H INX, D PUSH, RET,
5|SUBR RASETPT M E MOV, H INX, M D MOV, H INX, E VPTBL X STX,
6|D VPTBH X STX, RET,
7|SUBR RASETFP M E MOV, H INX, M D MOV, H INX, E VFNLPL X STX,
8|D VFNLPH X STX, RET,
9|SUBR RASCTXZW M A MOV, H INX, A VXZW X STX, RET,
10|SUBR RARANDOMDO M C MOV, H INX, M E MOV, H INX, M D MOV, H INX,
11|LDAR, C ANA, RNZ, XCHG, RET,
12|SUBR RASETXB M E MOV, H INX, M D MOV, H INX, E VFXBL X STX,
13|D VFXBH X STX, RET,
14|SUBR RASETYB M E MOV, H INX, M D MOV, H INX, E VFYBL X STX,
15|D VFYBH X STX, RET, -->

```

```

+-----Block      131-----
0|( JUMP TABLE FOR INTERPRETER ROUTINES )
1|DATA AJTBL RASETP , RASETM , RASETR , RAWAIT ,
2|RACALL , RAJMP , RASETDC , RASETDDC , RARET ,
3|RAHALT , RASETI , RASETXC , RASETYC , RADISP ,
4|RASETREP , RALOOP , RASETS , RAPATI , RASMCALL ,
5|RASETPT , RASETFP , RASCTXZW , RARANDOMDO ,
6|RASETXB , RASETYB , RASETXP ,
7|-->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      132-----
0|( ANIMATION UPDATOR ROUTINE )
1|F= ANIRET
2|SUBR ainter VATMR X A LDX, A ANA, RNZ, ( QUIT IF NOT NEEDED )
3|VPCL X L LDX, VPCH X H LDX,
4|LABEL ANIRET
5|ANIRET D LXI, D PUSH, M C MOV, H INX, 0 B MVI, XCHG,
6|AJTBL H LXI, B DAD, M C MOV, H INX, M B MOV, XCHG,
7|B PUSH, RET,
8|SUBR aup ainter CALL, PQSRH PQS X BITX, 0=, IF,
9|PQSUFH PQS X BITX, 0<,>, IF, VFNLPL X L LDX, VFNLPH X H LDX,
10|L VPATL X STX, H VPATH X STX, THEN, THEN, RET,
11|-->
12|
13|
14|
15|

```

```

+-----Block      133-----
0|( DECREMENT ANIMATION TIMERS, COMPUTE VECTORING TIME )
1|F= TBVD F= TBOK F= TBQUIT
2|SUBR TBCALC <ASSEMBLE DI,
3|PQTB X C LDX, VATMR X B LDX, B A MOV, C SUB,
4|0>=, IF, A VATMR X STX, 0 PQTB X MVIX,
5|ELSE, C A MOV, B SUB, A PQTB X STX, 0 VATMR X MVIX, B C MOV,
6|THEN, PQSNMT PQS X BITX, TBQUIT JRNZ, ( QUIT IF NO MASTER )
7|0 B MVI, VTLL X L LDX, VTLH X H LDX, A ANA, B DSBC,
8|TBVD JRZ, TBOK JP,
9|LABEL TBVD L A MOV, C ADD, A C MOV, A XRA, A H MOV, A L MOV,
10|A VATMR X STX, A PQTB X STX, PQSRH PQS X RESX,
11|LABEL TBOK L VTLL X STX, H VTLH X STX,
12|LABEL TBQUIT EI, RET, ASSEMBLE>
13|-->
14|
15|
+-----Block      134-----
0|( TIME BASED VECTOR UPDATE - IX=VECTOR ADDR, IY=QUEUE ENTRY )
1|( THIS VERSION VECTORS LINEARLY WITH LIMIT CHECKING )
2|HEX F= .VLP1 F= .VLP2 F= NUD
3|SUBR VECTLC <ASSEMBLE
4|C A MOV, A ANA, RZ, ( DONT IF ZERO VECTORING WANTED )
5|( NOW UPDATE COORDINATES )
6|VXL X L LDX, VXH X H LDX, VDXL X E LDX, VDXH X D LDX, C B MOV,
7|LABEL .VLP1 D DAD, .VLP1 DJNZ, H A MOV, 50 CPI, NUD JRNC,
8|L VXL X STX, H VXH X STX,
9|VYL X L LDX, VYH X H LDX, VDYL X E LDX, VDYH X D LDX, C B MOV,
10|LABEL .VLP2 D DAD, .VLP2 DJNZ, H A MOV, 0BA CPI, NUD JRNC,
11|L VYL X STX, H VYH X STX, 28 VXZW X MVIX, RET,
12|LABEL NUD PQSRH PQS X RESX, PQSDW PQS X SETX, RET, ASSEMBLE>
13|DECIMAL -->
14|
15|
+-----Block      135-----
0|( INITIALIZE INTERRUPT VERBS )
1|CC? NOT IFTRUE
2|LPINT IPNT 2 - U! TIMINT IPNT U! BGEND! BGINTVEC U!
3|: FIREUP INTSTART ; OTHERWISE
4|: FIREUP LPINT IPNT 2 - U! TIMINT IPNT U! BGEND! BGINTVEC U!
5|INTSTART ; IFEND
6|: START DI INIT INITFREELIST NILVQ LOCKOUTCOUNTER ZERO
7|BGTIMER ZERO LPFLAG ZERO 33 BGWINDOW ! 2 BGTLMT ! FIREUP ;
8|
9|( ROUTINE TO VWRITE WITH INTERCEPT CHECKING )
10|SUBR VIWRITE INTST IN, vwrite CALL, INTST IN,
11|A ANA, 0<>, IF, VIRL X L LDX, VIRH X H LDX,
12|H A MOV, L ORA, 0<>, IF, PCHL, THEN, THEN, RET,
13|DECIMAL
14|-->
15|

```

```

+-----Block      136-----
0|( SUBROUTINE TO UPDATE PATTERN USING XOR )
1|SUBR XAWRITE TBCALC CALL,
2|VECTLC CALL, ( UPDATE VECTOR )
3|POSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
4|POSDE PQS X RESX,
5| THEN, aup CALL, POSDW PQS X BITX, 0=,
6|IF, vwrite CALL, ELSE, POSDW PQS X RESX, POSDE PQS X SETX,
7|THEN, KILLOFF JMP,
8|( SUBROUTINE TO XAWRITE WITH INTERCEPT CHECKING )
9|SUBR XIWRITE TBCALC CALL,
10|VECTLC CALL, ( UPDATE DA VECTOR )
11|POSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
12|POSDE PQS X RESX, THEN, aup CALL, POSDW PQS X BITX, 0=,
13|IF, VIWRITE CALL, ELSE, POSDW PQS X RESX, POSDE PQS X SETX,
14|THEN, KILLOFF JMP,
15|-->

+-----Block      137-----
0|( SUBROUTINE TO VECTOR USING SECOND DERIVATIVE )
1|F= VUPX F= VUPY
2|SUBR VECTDD <ASSEMBLE POSFRZ PQS X BITX, 0<>, IF,
3|POSDW PQS X SETX, RET, THEN, C A MOV, A ANA, RZ, PSW PUSH,
4|VXL X L LDX, VXH X H LDX, VDXL X E LDX, VDXH X D LDX,
5|VDDXL X C LDX, VDDXH X B LDX,
6|LABEL VUPX XCHG, B DAD, XCHG, D DAD, A DCR, VUPX JRNZ,
7|E VDXL X STX, D VDXH X STX, L VXL X STX, H VXH X STX,
8|H A MOV, 80 CPI, CY~, IF, POSFRZ PQS X SETX, POSDW PQS X SETX,
9|THEN, PSW POP, VYL X L LDX, VYH X H LDX, VDYL X E LDX,
10|VDYH X D LDX, VDDYL X C LDX, VDDYH X B LDX,
11|LABEL VUPY XCHG, B DAD, XCHG, D DAD, A DCR, VUPY JRNZ,
12|E VDYL X STX, D VDYH X STX, L VYL X STX, H VYH X STX,
13|H A MOV, 182 CPI, RC, POSFRZ PQS X SETX, POSDW PQS X SETX, RET,
14|ASSEMBLE> -->
15|

+-----Block      138-----
0|( SUBROUTINE TO UPDATE PATTERN USING XOR AND 2ND DERV VECTOR )
1|SUBR XADDDWRITE TBCALC CALL,
2|VECTDD CALL,
3|POSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
4|POSDE PQS X RESX,
5| THEN, aup CALL,
6|POSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE,
7|POSDE PQS X SETX, POSDW PQS X RESX, THEN, KILLOFF JMP,
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      139-----
0|( UPDATE VECTOR FROM JOYSTICK ) HEX 11 C= JOYSTICK
1|F= JYLP1 F= JYLP2 F= JXLP1 F= JXLP2 F= JXCK
2|SUBR JOYUPD <ASSEMBLE C A MOV, A ANA, RZ, JOYSTICK IN, 18 ANI,
3|18 CPI, JXCK JZ, VYL X L LDX, VYH X H LDX, VDYL X E LDX,
4|VDYH X D LDX, C B MOV, JOYSTICK IN, 10 ANI, 0<>, IF,
5|LABEL JYLP1 A ANA, D DSBC, JYLP1 DJNZ, H A MOV, VDDYL X CMPX,
6|CY, IF, VDDYL X H LDX, 0 L MVI, THEN, ELSE,
7|LABEL JYLP2 D DAD, JYLP2 DJNZ, H A MOV, VDDYH X CMPX, CY~, IF,
8|VDDYH X H LDX, 0 L MVI, THEN, THEN, L VYL X STX, H VYH X STX,
9|LABEL JXCK JOYSTICK IN, 6 ANI, 6 CPI, RZ, VXL X L LDX,
10|VXH X H LDX, VDXL X E LDX, VDXH X D LDX, C B MOV, JOYSTICK IN,
11|4 ANI, 0=, IF, LABEL JXLP1 A ANA, D DSBC, JXLP1 DJNZ, H A MOV,
12|VDDXL X CMPX, CY, IF, VDDXL X H LDX, 0 L MVI, THEN, ELSE,
13|LABEL JXLP2 D DAD, JXLP2 DJNZ, H A MOV, VDDXH X CMPX, CY~, IF,
14|VDDXH X H LDX, 0 L MVI, THEN, THEN, L VXL X STX, H VXH X STX,
15|RET, ASSEMBLE> DECIMAL -->
+-----Block      140-----
0|( SUBROUTINE TO UPDATE PATTERN FROM JOYSTICK )
1|SUBR JOYWRITE TBCALC CALL,
2|JOYUPD CALL, ( UPDATE FROM JOYSTICK )
3|38 VXZW X MVIX,
4|PQSD PQS X BITX, 0=, IF, verase CALL, ELSE,
5|PQSD PQS X RESX,
6| THEN, aup CALL, PQSDW PQS X BITX, 0=,
7|IF, VIWRITE CALL,
8|ELSE, PQSD PQS X SETX, PQSDW PQS X RESX, THEN, KILLOFF JMP,
9|-->
10|
11|
12|
13|
14|
15|
+-----Block      141-----
0|( COMPUTE DELTA FOR 1 COORDINATE )
1|( FIRST A NEGATION SUBROUTINE )
2|SUBR CMPHL H A MOV, CMA, A H MOV, L A MOV, CMA, A L MOV, H INX,
3|RET,
4|( IN: HL=TARGET, DE=TIME, BC=START )
5|SUBR CDELTA B PUSH, A ANA, B DSBC, CY~, IF, UNSDIV CALL,
6|ELSE, CMPHL CALL, UNSDIV CALL, CMPHL CALL, XCHG, CMPHL CALL,
7|XCHG, THEN, B POP, B DAD, RET,
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      142-----
0|( CLEAR VECTOR ) F= INIZL
1|SUBR CLRVEC <ASSEMBLE X PUSHX, H POP, 64 B MVI, A XRA,
2|LABEL INIZL A M MOV, H INX, INIZL DJNZ, RET, ASSEMBLE>
3|( RESET ANIMATION STUFF )
4|SUBR CRASHA DI, L VPCL X STX, H VPCH X STX,
5|X PUSHX, H POP, VASTKS D LXI, D DAD, L VSPL X STX,
6|H VSPH X STX, 0 VATMR X MVIX, EI, RET,
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      143-----
0|( SUBROUTINE TO PUT VECTOR ON PROCESS Q )
1|SUBR STARTVEC DI, Y PUSHX,
2|X PUSHX, H POP, 0 vahead Y LXIX, ADDQ CALL,
3|Y POPX, EI, RET,
4|( SUBROUTINE TO INITIALIZE A STANDARD XOR WRITE )
5|HEX
6|SUBR SETSTDW 8 Y L LDX, 9 Y H LDX, CRASHA CALL,
7|6 Y L LDX, 7 Y H LDX, L VTLL X STX, H VTLH X STX, L VRACK X STX,
8| 5 Y A LDX, A VIDENT X STX, ( SET ID BYTE WITH H.O. STATUS )
9|4 Y A LDX, A PQS X STX, 20 VMAGIC X MVIX,
10|30 VXZW X MVIX, XAWRITE H LXI,
11|L PQLR X STX, H PQRH X STX, RET,
12|DECIMAL -->
13|
14|
15|

+-----Block      144-----
0|( XVMOVE COMMAND - MOVE AN EXISTING VECTOR )
1|( VRACK X1 Y1 X2 Y2 ALIST TIME STATUS VECTOR VMOVE )
2|CODE XVMOVE X PUSHX, H POP, Y PUSHX, D POP, EXX,
3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX, CLRVEC CALL,
4|16 Y C LDX, 17 Y B LDX, 12 Y L LDX, 13 Y H LDX,
5|6 Y E LDX, 7 Y D LDX, D PUSH, CDELTA CALL,
6|L VXL X STX, H VXH X STX, E VDXL X STX, D VDXH X STX,
7|D POP, 14 Y C LDX, 15 Y B LDX, 10 Y L LDX, 11 Y H LDX,
8|CDELTA CALL, L VYL X STX, H VYH X STX, E VDYL X STX,
9|D VDYH X STX, SETSTDW CALL, 18 Y A LDX, A VRACK X STX,
10|STARTVEC CALL, UNFRAME 18 H LXI, SP DAD, SPHL,
11|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT
12|: VMOVE GETNODE DUP 0 <> IF XVMOVE ELSE XDI ." FUBAR" THEN ;
13|DECIMAL -->
14|
15|

```

```

+-----Block      145-----
0|( XSTART COMMAND - START AN EXISTING VECTOR )
1|( ALIST TIME STATUS VECTOR VMOVE )
2|CODE XVSTART X PUSHX, H POP, Y PUSHX, D POP, EXX,
3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX, CLRVEC CALL,
4|SETSTDW CALL, STARTVEC CALL,
5|UNFRAME 8 H LXI, SP DAD, SPHL,
6|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT
7|: VSTART GETNODE DUP 0 < > IF XVSTART ELSE XDI ." FUBAR" THEN ;
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      146-----
0|( START A VECTOR WITH JUST INITIAL X AND Y ) DECIMAL
1|( X Y ANIMATION TIME STATUS XYVSTART )
2|SUBR XYVSTART DI, getnode CALL, EI, H PUSH,
3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX,
4|CLRVEC CALL, 6 Y C LDX, C VRACK X STX,
5|12 Y L LDX, 13 Y H LDX, L VXL X STX, H VXH X STX,
6|10 Y L LDX, 11 Y H LDX, L VYL X STX, H VYH X STX,
7|SETSTDW CALL, STARTVEC CALL,
8|UNFRAME 12 H LXI, SP DAD, SPHL, RET,
9|( CLUDGE TO CALL AS A VERB - VERY BIZARRE BUT IT SHOULD WORK )
10|SUBR CLUDGEIT H PUSH, D PUSH, B PUSH, EXX, H PUSH, D PUSH,
11|X PUSHX, D POP, Y PUSHX, H POP, EXX, XYVSTART JMP,
12|( X Y ANIMATION TIME STATUS XYVECTOR )
13|CODE XYVECTOR D POP, H POP, EXX, B POP, D POP, H POP,
14|CLUDGEIT CALL, EXX, D PUSH, X POPX, H PUSH, Y POPX, NEXT
15|-->

```

```

+-----Block      147-----
0|( CHECK FOR INTERCEPT WITH VECTOR )
1|F= NOINT
2|SUBR CHECKVEC <ASSEMBLE
3|PQSRH PQS Y BITX, NOINT JZ, ( IF DEAD ALREADY )
4|PQSDE PQS Y BITX, NOINT JNZ, ( OR IF NOT WRITTEN )
5|VPATL X L LDX, VPATH X H LDX, M C MOV, H INX, M E MOV,
6|VPATL Y L LDX, VPATH Y H LDX, M B MOV, H INX, M D MOV,
7|VYH X A LDX, VYH Y SUBX, 0>=, IF, D CMP,
8|NOINT JNC, ELSE, E ADD, NOINT JM, THEN,
9|VXH X A LDX, VXH Y SUBX,
10|0>=, IF, B CMP,
11|NOINT JNC, ELSE, C ADD,
12|NOINT JM, THEN, ( A XVI, A ANA, RET,
13|LABEL NOINT A XRA, RET, ASSEMBLE>
14|-->
15|

```



```

+-----Block      148-----
0|( CHECK GROUP OF VECTORS FOR INTERCEPT )
1|( IX = ME, C=MASK SPECIFYING SUBSET TO EXAMINE )
2|( RETURNS NZ AND IY=CULPRIT IF FOUND, ELSE Z )
3|F= ICKL F= NOTEND F= NOTHIM
4|SUBR CHECKALL <ASSEMBLE
5|0 vqhead LHLD, ( HL=NEXT FELLA AFTER ME )
6|LABEL ICKL X PUSHX, D POP, ( DE=ME )
7|H A MOV, D CMP, NOTEND JRNZ, L A MOV, E CMP, RZ,
8|LABEL NOTEND H PUSH, Y POPX,
9|VIDENT Y A LDX, C ANA, NOTHIM JRZ, ( SELECTED BY MASK? )
10|B PUSH, CHECKVEC CALL, B POP,
11|RNZ, ( KICKOUT IF HE IS IT )
12|LABEL NOTHIM PQFL Y L LDX, PQFH Y H LDX, ICKL JMPR,
13|ASSEMBLE>
14|DECIMAL
15|-->

```

```

+-----Block      149-----
0|( NUMBER PATTERNS , 5 X 7 ORDERED 0-9 )
1|CC? IFTRUE ROMIT IFEND
2|DATA NUMPATS BINARY
3|011111100 B, 100000010 B, 100000010 B, 100000010 B, 011111100 B,
4|000000000 B, 100000100 B, 111111110 B, 100000000 B, 000000000 B,
5|111100100 B, 10010010 B, 10010010 B, 10010010 B, 10001100 B,
6|01000100 B, 10000010 B, 10010010 B, 10010010 B, 01101100 B,
7|00110000 B, 00101000 B, 00100100 B, 11111110 B, 00100000 B,
8|01001110 B, 10001010 B, 10001010 B, 10001010 B, 01110010 B,
9|011111000 B, 10010100 B, 10010010 B, 10010010 B, 01100000 B,
10|00000010 B, 11100010 B, 00010010 B, 00001010 B, 00000110 B,
11|01101100 B, 10010010 B, 10010010 B, 10010010 B, 01101100 B,
12|00001100 B, 10010010 B, 10010010 B, 01010010 B, 00111100 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      150-----
0|( ROUTINE TO DISPLAY A BCD NUMBER 3 DIGITS LONG FROM VECTOR )
1|HEX F= SHOWNUM F= NUMLP
2|SUBR DISPBCD3 <ASSEMBLE
3|VXL X E LDX, VXH X D LDX, VYL X L LDX, VYH X H LDX,
4|428 B LXI, relabs CALL, XCHG, C A MOV,
5|MAGIC OUT, B A MOV, XPAND OUT,
6|VPATL X L LDX, VPATH X H LDX, H INX, M A MOV, OF ANI,
7|SHOWNUM CNZ, H DCX, M A MOV, RRC, RRC, RRC, RRC,
8|SHOWNUM CALL, M A MOV,
9|LABEL SHOWNUM H PUSH, F ANI, A C MOV, RLC, RLC, C ADD, A C MOV,
10|0 B MVI, NUMPATS H LXI, B DAD, XCHG, E B MVI,
11|LABEL NUMLP D LDAX, A M MOV, H INX, A M MOV, A XRA, H INX,
12|A M MOV, H INX, A M MOV, D INX,
13|L A MOV, 4D ADI, A L MOV, 0 A MVI, H ADC, A H MOV,
14|NUMLP DJNZ, 50 D LXI, D DAD, XCHG, H POP, RET, ASSEMBLE>
15|DECIMAL -->

```

```

+-----Block      151-----
0|( INTERRUPT WRITE NUMBER ROUTINE )
1|SUBR NUMWRITE
2|TBCALC CALL,
3|aup CALL,
4|PQSDW PQS X BITX, 0=, IF, DISPBCD3 CALL, THEN,
5|KILLOFF JMP,
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      152-----
0|( BASE STATION )
1|DECIMAL DATA FIREBASE 5 B, 13 B,
2|QUAD 0222 B, 2222 B, 2000 B, 0000 B, 0 B,
3|2111 B, 1111 B, 2220 B, 0000 B, 0 B,
4|0222 B, 2222 B, 2000 B, 0000 B, 0 B,
5|0000 B, 1110 B, 0022 B, 2200 B, 0 B,
6|0000 B, 0111 B, 0002 B, 2220 B, 0 B,
7|1111 B, 1111 B, 1102 B, 2222 B, 0 B,
8|0002 B, 2222 B, 2222 B, 2222 B, 1000 B,
9|1111 B, 1111 B, 1102 B, 2222 B, 0 B,
10|0000 B, 0111 B, 0002 B, 2220 B, 0 B,
11|0000 B, 1110 B, 0022 B, 2200 B, 0 B,
12|0222 B, 2222 B, 2000 B, 0000 B, 0 B,
13|2111 B, 1111 B, 2220 B, 0000 B, 0 B,
14|0222 B, 2222 B, 2000 B, 0000 B, 0 B, -->
15|
+-----Block      153-----
0|( SMALL BASE ) DECIMAL DATA SMALBASE 4 B, 11 B, QUAD
1|0222 B, 2220 B, 0000 B, 0 B,
2|2222 B, 2200 B, 0000 B, 0 B,
3|0011 B, 0000 B, 0000 B, 0 B,
4|0111 B, 1000 B, 2200 B, 0 B,
5|1111 B, 1110 B, 0220 B, 0 B,
6|0002 B, 2222 B, 2222 B, 2000 B,
7|1111 B, 1110 B, 0220 B, 0 B,
8|0111 B, 1000 B, 2200 B, 0 B,
9|0011 B, 0000 B, 0000 B, 0 B,
10|2222 B, 2200 B, 0000 B, 0 B,
11|0222 B, 2220 B, 0000 B, 0 B,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      154-----
0|( GORF ) DECIMAL DATA GORF 6 B, 15 B, QUAD
1|2000 B, 3330 B, 0000 B, 0000 B, 0010 B, 0 B,
2|2003 B, 3333 B, 3300 B, 0000 B, 0100 B, 0 B,
3|2223 B, 3333 B, 3333 B, 3000 B, 1000 B, 0 B,
4|0003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0 B,
5|0033 B, 3331 B, 1111 B, 3333 B, 3300 B, 0 B,
6|0031 B, 1132 B, 2111 B, 3333 B, 3330 B, 0 B,
7|0032 B, 1233 B, 3333 B, 3333 B, 3333 B, 0 B,
8|0031 B, 1133 B, 3333 B, 3333 B, 3333 B, 0 B,
9|0032 B, 1233 B, 3333 B, 3333 B, 3333 B, 0 B,
10|0031 B, 1131 B, 1111 B, 3333 B, 3330 B, 0 B,
11|0033 B, 3332 B, 2111 B, 3333 B, 3300 B, 0 B,
12|0003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0 B,
13|2223 B, 3333 B, 3333 B, 3000 B, 1000 B, 0 B,
14|2003 B, 3333 B, 3300 B, 0000 B, 0100 B, 0 B,
15|2000 B, 3330 B, 0000 B, 0000 B, 0010 B, 0 B, -->

```

```

+-----Block      155-----
0|( GORFB ) DECIMAL DATA GORFB 6 B, 15 B, QUAD
1|0000 B, 3330 B, 0000 B, 0000 B, 2000 B, 0000 B,
2|0003 B, 3333 B, 3300 B, 0000 B, 0200 B, 0000 B,
3|1113 B, 3333 B, 3333 B, 3000 B, 2000 B, 0000 B,
4|1003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0000 B,
5|1033 B, 3331 B, 1222 B, 3333 B, 3300 B, 0000 B,
6|0033 B, 2332 B, 2222 B, 3333 B, 3330 B, 0000 B,
7|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
8|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
9|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
10|0033 B, 2331 B, 1222 B, 3333 B, 3330 B, 0000 B,
11|1033 B, 3332 B, 2222 B, 3333 B, 3300 B, 0000 B,
12|1003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0000 B,
13|1113 B, 3333 B, 3333 B, 3000 B, 2000 B, 0000 B,
14|0003 B, 3333 B, 3300 B, 0000 B, 0200 B, 0000 B,
15|0000 B, 3330 B, 0000 B, 0000 B, 2000 B, 0000 B, DECIMAL -->

```

```

+-----Block      156-----
0|( GORF 2 AND GORF 3 )
1|DECIMAL DATA GORF2 3 B, 5 B, QUAD
2|~ 2033 0010 0000 ^
3|~ 2333 1300 0000 ^
4|~ 0313 3330 0000 ^
5|~ 2333 1300 0000 ^
6|~ 2033 0010 0000 ^
7|DECIMAL DATA GORF2 3 B, 7 B, QUAD
8|~ 2003 3300 1000 ^
9|~ 2233 3333 0000 ^
10|~ 0032 3213 3000 ^
11|~ 0332 3333 3000 ^
12|~ 0032 3213 3000 ^
13|~ 2233 3333 0000 ^
14|~ 2003 3300 1000 ^
15|DECIMAL -->

```

```

+-----Block      157-----
0|( GORF 1 AND GORF 4 )
1|DATA GORF1 2 B, 3 B, QUAD
2|~ 2033 0000 ^
3|~ 0313 3000 ^
4|~ 2033 0000 ^
5|DECIMAL DATA GORF4 4 B, 9 B, QUAD
6|~ 2003 3300 0001 0000 ^
7|~ 2033 3333 0010 0000 ^
8|~ 2233 3333 3300 0000 ^
9|~ 0032 3211 3330 0000 ^
10|~ 0031 3333 3330 0000 ^
11|~ 0032 3211 3330 0000 ^
12|~ 2233 3333 3300 0000 ^
13|~ 2033 3333 0010 0000 ^
14|~ 2003 3300 0001 0000 ^
15|DECIMAL -->

```

```

+-----Block      158-----
0|( GORF 5 )
1|DATA GORF5 5 B, 12 B, QUAD
2|~ 2000 3333 0000 0010 0000 ^
3|~ 2003 3333 3330 0100 0000 ^
4|~ 2223 3333 3333 3000 0000 ^
5|~ 0003 3332 2113 3300 0000 ^
6|~ 0033 1231 1113 3330 0000 ^
7|~ 0033 1133 3333 3333 0000 ^
8|~ 0033 2133 3333 3333 0000 ^
9|~ 0033 1232 2113 3330 0000 ^
10|~ 0003 3331 1113 3300 0000 ^
11|~ 2223 3333 3333 3000 0000 ^
12|~ 2003 3333 3330 0100 0000 ^
13|~ 2000 3333 0000 0010 0000 ^
14|DECIMAL -->
15|

```

```

+-----Block      159-----
0|( FIRE BASE EXPLOSION PATTERN )
1|DATA FBEXP1 4 B, 12 B, QUAD
2|0000 B, 3000 B, 0010 B, 0000 B,
3|3000 B, 3303 B, 0000 B, 0000 B,
4|0333 B, 3333 B, 0030 B, 0000 B,
5|0033 B, 3333 B, 3330 B, 0000 B,
6|0033 B, 3133 B, 1333 B, 0000 B,
7|3333 B, 1111 B, 1330 B, 0000 B,
8|3331 B, 1111 B, 1000 B, 0000 B,
9|0333 B, 3311 B, 1030 B, 0000 B,
10|0033 B, 0333 B, 3000 B, 0000 B,
11|1030 B, 0033 B, 3000 B, 0000 B,
12|0000 B, 0033 B, 3000 B, 0000 B,
13|0000 B, 0003 B, 0000 B, 0000 B,
14|DECIMAL -->
15|

```

```

+-----Block      160-----
0|( ANOTHER FIREBASE EXPLOSION PATTERN )
1| DATA FBEXP2 5 B, 17 B, QUAD
2| 0001 B, 0000 B, 0000 B, 0000 B, 0000 B,
3| 0000 B, 1000 B, 0000 B, 0000 B, 0000 B,
4| 0000 B, 1110 B, 0000 B, 0000 B, 0000 B,
5| 0000 B, 0110 B, 0010 B, 0000 B, 0000 B,
6| 0000 B, 0111 B, 1110 B, 0000 B, 3000 B,
7| 0000 B, 0131 B, 1110 B, 0110 B, 0000 B,
8| 0000 B, 1133 B, 3111 B, 1111 B, 0000 B,
9| 0000 B, 1133 B, 3333 B, 3311 B, 0000 B,
10| 3001 B, 1333 B, 3333 B, 3310 B, 0000 B,
11| 0111 B, 1333 B, 3333 B, 3110 B, 0000 B,
12| 0111 B, 3333 B, 3333 B, 3110 B, 0000 B,
13| 0011 B, 3333 B, 3333 B, 3311 B, 0000 B,
14| 0011 B, 3311 B, 3331 B, 3310 B, 0000 B,
15|-->

```

```

+-----Block      161-----
0|( CONTINUATION OF FBEXP2, PHASOR AND NULPAT )
1| 0000 B, 1111 B, 1111 B, 1110 B, 0000 B,
2| 0011 B, 0110 B, 0110 B, 1011 B, 1000 B,
3| 0010 B, 0000 B, 0000 B, 1000 B, 1000 B,
4| 1100 B, 0000 B, 0000 B, 1000 B, 0000 B,
5| DECIMAL DATA FBURST 4 B, 1 B, QUAD 1111 B, 1111 B, 1111 B, 0 B,
6| DECIMAL DATA NULPAT 1 B, 1 B, 0 B,
7| DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      162-----
0|( FBEXP3 )
1| DATA FBEXP3 6 B, 23 B, QUAD
2| 0000 B, 0000 B, 2000 B, 0000 B, 0000 B, 0000 B,
3| 0000 B, 0002 B, 1200 B, 0000 B, 0000 B, 0000 B,
4| 0000 B, 2221 B, 1120 B, 0000 B, 0000 B, 0000 B,
5| 0002 B, 1111 B, 1112 B, 0022 B, 0000 B, 0000 B,
6| 0023 B, 3133 B, 3311 B, 2212 B, 2000 B, 0000 B,
7| 0211 B, 3333 B, 3331 B, 1331 B, 1200 B, 0000 B,
8| 0021 B, 1333 B, 3333 B, 3331 B, 1120 B, 0000 B,
9| 0002 B, 3333 B, 3333 B, 3333 B, 1112 B, 0000 B,
10| 0000 B, 2333 B, 3333 B, 3333 B, 3111 B, 2000 B,
11| 0002 B, 1133 B, 3333 B, 3333 B, 3331 B, 1200 B,
12| 0021 B, 1333 B, 3333 B, 3333 B, 3311 B, 2000 B,
13| 0211 B, 3333 B, 3333 B, 3333 B, 3112 B, 0000 B,
14| 2211 B, 3333 B, 3333 B, 3333 B, 1120 B, 0000 B,
15|-->

```

```

+-----Block      163-----
0|( CONTINUED FBEXP3 )
1|0211 B, 1133 B, 3333 B, 3311 B, 1120 B, 0000 B,
2|0021 B, 3333 B, 3333 B, 3331 B, 2200 B, 0000 B,
3|0002 B, 3323 B, 3333 B, 3332 B, 0000 B, 0000 B,
4|0000 B, 3231 B, 3333 B, 3332 B, 0000 B, 0000 B,
5|0000 B, 2133 B, 3333 B, 3332 B, 2200 B, 0000 B,
6|0002 B, 1113 B, 3133 B, 3331 B, 1120 B, 0000 B,
7|0000 B, 2111 B, 1123 B, 1311 B, 1200 B, 0000 B,
8|0000 B, 0211 B, 1202 B, 1111 B, 2000 B, 0000 B,
9|0000 B, 0021 B, 2000 B, 2112 B, 0000 B, 0000 B,
10|0000 B, 0002 B, 0000 B, 0220 B, 0000 B, 0000 B,
11|-->
12|
13|
14|
15|

```

```

+-----Block      164-----
0|( FBEXP4 )
1|DECIMAL DATA FBEXP4 6 B, 23 B, QUAD
2|2000 B, 0000 B, 0020 B, 0000 B, 0000 B, 2000 B,
3|0200 B, 0000 B, 0200 B, 0000 B, 0002 B, 0000 B,
4|0020 B, 0000 B, 2000 B, 0000 B, 0020 B, 0000 B,
5|0002 B, 0000 B, 2000 B, 0000 B, 0200 B, 0000 B,
6|0000 B, 2202 B, 2201 B, 1000 B, 2000 B, 0000 B,
7|0000 B, 0223 B, 1332 B, 1102 B, 0000 B, 0000 B,
8|0000 B, 2233 B, 3333 B, 2110 B, 0002 B, 0000 B,
9|0000 B, 0231 B, 3333 B, 3310 B, 0020 B, 0000 B,
10|0000 B, 0223 B, 1333 B, 3111 B, 2200 B, 0000 B,
11|0000 B, 0023 B, 1333 B, 3120 B, 0000 B, 0000 B,
12|0000 B, 2223 B, 1333 B, 3122 B, 0000 B, 0000 B,
13|0002 B, 2331 B, 3333 B, 3312 B, 0000 B, 0000 B,
14|0022 B, 3133 B, 3333 B, 3220 B, 0000 B, 0000 B,
15|-->

```

```

+-----Block      165-----
0|( FBEXP4 CONTINUED )
1|0223 B, 3113 B, 3333 B, 1200 B, 0000 B, 0000 B,
2|0002 B, 2311 B, 3331 B, 1220 B, 0000 B, 0000 B,
3|0002 B, 2233 B, 3331 B, 3322 B, 0000 B, 0000 B,
4|0020 B, 0232 B, 2233 B, 3320 B, 0000 B, 0000 B,
5|0200 B, 0220 B, 0223 B, 3222 B, 0000 B, 0000 B,
6|2000 B, 0000 B, 0022 B, 2202 B, 2000 B, 0000 B,
7|0000 B, 0000 B, 0020 B, 0000 B, 0200 B, 0000 B,
8|0000 B, 0000 B, 0200 B, 0000 B, 0020 B, 0000 B,
9|0000 B, 0000 B, 2000 B, 0000 B, 0002 B, 0000 B,
10|0000 B, 0000 B, 0000 B, 0000 B, 0000 B, 2000 B,
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      166-----
0|( FIREBASE EXPLOSION 5 )
1|DATA FBEXP5 6 B, 23 B, QUAD
2|~ 0000 0000 0010 0003 0000 0000 ^
3|~ 0000 0300 1001 1000 0300 0000 ^
4|~ 0000 2030 0200 0020 0000 0000 ^
5|~ 0000 0000 0001 0000 0000 0000 ^
6|~ 0300 2100 2000 3000 0200 0000 ^
7|~ 0030 0200 0002 0000 0033 3000 ^
8|~ 0300 0020 0220 0030 0300 0000 ^
9|~ 0000 0000 0300 0000 0003 0000 ^
10|~ 0033 3002 2200 0110 0011 1000 ^
11|~ 0003 3001 2000 1100 0122 0000 ^
12|~ 0033 2201 2201 1010 1022 1000 ^
13|~ 3300 0221 2011 1100 1110 0000 ^
14|~ 3330 0022 2001 1110 1100 0000 ^
15|~ 0030 2000 0222 2100 0000 1000 ^ -->

```

```

+-----Block      167-----
0|( FBEXP5 CONTINUED )
1|~ 0033 1110 0020 1020 1100 0000 ^
2|~ 0300 0110 0200 1000 0100 0000 ^
3|~ 0001 0003 0030 0010 0030 0000 ^
4|~ 0000 0100 0000 0000 1000 0000 ^
5|~ 0200 0000 0010 0000 0000 0000 ^
6|~ 0001 0030 0000 0030 0010 0000 ^
7|~ 0000 0000 0020 0000 0000 0000 ^
8|~ 0000 0010 0000 0100 3000 0000 ^
9|~ 0000 0000 0102 0030 0000 0000 ^

```

```

10|DECIMAL -->

```

```

11|

```

```

12|

```

```

13|

```

```

14|

```

```

15|

```

```

+-----Block      168-----
0|( FIRE BASE EXPLOSION 6 )
1|DATA FBEXP6 6 B, 23 B, QUAD
2|~ 0000 0000 3000 0000 0000 0000 ^
3|~ 0000 0020 0000 0300 0000 0000 ^
4|~ 0000 0000 0300 0000 0010 0000 ^
5|~ 0001 0000 0000 0200 0000 0000 ^
6|~ 0000 0000 0000 0000 0000 0000 ^
7|~ 0002 0010 0031 0001 0200 0000 ^
8|~ 0002 0000 1000 0000 0000 0000 ^
9|~ 0000 0300 0001 0020 0000 3000 ^
10|~ 0100 0000 0000 2000 0000 0000 ^
11|~ 0001 0002 0200 0001 0300 0000 ^
12|~ 3000 0000 0020 0000 0003 0000 ^
13|~ 0020 0200 0000 0030 0010 0000 ^
14|~ 0000 0001 0202 0001 2000 0000 ^
15|~ 2030 1000 0000 0203 0000 0000 ^ -->

```

```

+-----Block      169-----
0|( FIRE BASE EXPLOSION 6 CONTINUED )
1|~ 0001 0000 0200 0003 0000 1000 ^
2|~ 1000 0010 0003 0100 0020 0000 ^
3|~ 0000 0000 1000 0000 0000 0000 ^
4|~ 0030 0300 0000 0000 0200 0000 ^
5|~ 0000 0000 0000 0000 0000 0000 ^
6|~ 0000 3010 0000 1000 0200 0000 ^
7|~ 0000 0100 0200 0003 0000 0000 ^
8|~ 0000 0003 0010 0000 0000 0000 ^
9|~ 0000 0001 0020 1000 0000 0000 ^
10|DECIMAL -->
11|
12|
13|
14|
15|
+-----Block      170-----
0|( ALIEN EXPLOSION PATTERN )
1|DATA EXPLOSION1 3 B, 11 B, QUAD
2|0000 B, 0000 B, 0000 B, 0000 B, 0000 B, 0000 B,
3|0000 B, 0010 B, 0000 B, 0100 B, 3000 B, 0000 B,
4|0033 B, 3330 B, 0000 B, 0003 B, 1300 B, 0000 B,
5|0031 B, 1130 B, 0000 B, 0033 B, 1330 B, 0000 B,
6|0103 B, 3000 B, 0000 B, 0000 B, 0010 B, 0000 B,
7|0000 B, 0000 B, 0000 B,
8|DECIMAL DATA EXPLOSION2 3 B, 11 B, QUAD
9|1001 B, 0001 B, 0000 B, 0100 B, 1010 B, 0000 B,
10|0011 B, 1100 B, 0000 B, 0111 B, 3111 B, 0000 B,
11|0013 B, 3310 B, 0000 B, 1011 B, 3110 B, 0000 B,
12|0113 B, 1100 B, 0000 B, 0011 B, 0111 B, 0000 B,
13|0101 B, 0110 B, 0000 B, 0000 B, 0010 B, 0000 B,
14|1000 B, 1001 B, 0000 B,
15|DECIMAL -->
+-----Block      171-----
0|( MORE ALIEN EXPLOSIONS )
1|DATA EXPLOSION3 3 B, 12 B, QUAD
2|0000 B, 0101 B, 0000 B,
3|1000 B, 0000 B, 0000 B,
4|0010 B, 0000 B, 0000 B,
5|0000 B, 3000 B, 0000 B,
6|0003 B, 3301 B, 0000 B,
7|0033 B, 2301 B, 0000 B,
8|0003 B, 2300 B, 0000 B,
9|0003 B, 3330 B, 0000 B,
10|0100 B, 3000 B, 0000 B,
11|0000 B, 0000 B, 0000 B,
12|0001 B, 0000 B, 0000 B,
13|0000 B, 0010 B, 0000 B,
14|DECIMAL -->
15|

```



```

+-----Block      172-----
0|( EXPLOSION PATTERNS ) DECIMAL
1|DATA EXPLOSION4 3 B, 11 B, QUAD
2|3001 B, 0020 B, 0000 B, 3001 B, 0030 B, 0000 B,
3|0300 B, 0300 B, 0000 B, 0001 B, 1003 B, 0000 B,
4|0111 B, 1100 B, 0000 B, 2110 B, 1102 B, 0000 B,
5|0101 B, 1000 B, 0000 B, 0111 B, 1101 B, 0000 B,
6|2001 B, 0000 B, 0000 B, 3001 B, 0220 B, 0000 B,
7|0020 B, 0101 B, 0000 B, DECIMAL
8|DATA EXPLOSION5 3 B, 11 B, QUAD
9|0000 B, 0010 B, 0000 B, 0200 B, 0000 B, 0000 B,
10|0000 B, 0300 B, 0000 B, 0000 B, 0000 B, 0000 B,
11|2000 B, 0010 B, 0000 B, 0030 B, 0012 B, 0000 B,
12|0000 B, 1000 B, 0000 B, 0100 B, 0000 B, 0000 B,
13|0020 B, 2030 B, 0000 B, 3000 B, 0000 B, 0000 B,
14|0002 B, 0001 B, 0000 B, DECIMAL
15|CC? IFTRUE TIMOR IFEND -->

```

```

+-----Block      173-----
0|( KAMIZAKE PATTERN )
1|DECIMAL DATA KAMI 3 B, 11 B, QUAD
2|0000 B, 0001 B, 0000 B,
3|0000 B, 0011 B, 0000 B,
4|0000 B, 0201 B, 0000 B,
5|0000 B, 2000 B, 0000 B,
6|3301 B, 1100 B, 0000 B,
7|3011 B, 1110 B, 0000 B,
8|3301 B, 1100 B, 0000 B,
9|0000 B, 2000 B, 0000 B,
10|0000 B, 0201 B, 0000 B,
11|0000 B, 0011 B, 0000 B,
12|0000 B, 0001 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      174-----
0|( ROTATED KAMIKAZE 1 )
1|DECIMAL DATA KMKZ1R 4 B, 10 B, QUAD
2|0000 B, 0100 B, 0000 B, 0000 B,
3|0000 B, 0110 B, 0000 B, 0000 B,
4|0000 B, 2110 B, 0000 B, 0000 B,
5|0000 B, 2000 B, 0000 B, 0000 B,
6|0000 B, 1110 B, 0200 B, 0000 B,
7|0301 B, 1110 B, 0000 B, 0000 B,
8|3011 B, 1100 B, 0000 B, 0000 B,
9|3301 B, 1110 B, 1000 B, 0000 B,
10|0000 B, 0221 B, 1100 B, 0000 B,
11|0000 B, 0000 B, 0100 B, 0000 B,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      175-----
0|( ROTATED KAMIKAZE 2 )
1|DECIMAL DATA KMKZ2R 4 B, 10 B, QUAD
2|0010 B, 0000 B, 0000 B, 0000 B,
3|0011 B, 0000 B, 0000 B, 0000 B,
4|0010 B, 0000 B, 0000 B, 0000 B,
5|0020 B, 0000 B, 0000 B, 0000 B,
6|0021 B, 0100 B, 0000 B, 0000 B,
7|0011 B, 1000 B, 0000 B, 0000 B,
8|0111 B, 1100 B, 1000 B, 0000 B,
9|0011 B, 1221 B, 1100 B, 0000 B,
10|3101 B, 0000 B, 0000 B, 0000 B,
11|3300 B, 0000 B, 0000 B, 0000 B,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      176-----
0|( ROTATED KAMIKAZE 3 )
1|DECIMAL DATA KMKZ3R 4 B, 10 B, QUAD
2|1100 B, 0000 B, 0000 B, 0000 B,
3|0110 B, 0000 B, 0000 B, 0000 B,
4|0100 B, 0000 B, 0000 B, 0000 B,
5|0210 B, 1101 B, 1000 B, 0000 B,
6|0211 B, 1101 B, 1100 B, 0000 B,
7|0011 B, 1122 B, 0000 B, 0000 B,
8|0011 B, 1000 B, 0000 B, 0000 B,
9|0001 B, 0000 B, 0000 B, 0000 B,
10|0030 B, 3000 B, 0000 B, 0000 B,
11|0033 B, 0000 B, 0000 B, 0000 B,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      177-----
0|( ROTATED KAMIKAZE 4 )
1|DECIMAL DATA KMKZ4R 4 B, 8 B, QUAD
2|1110 B, 0000 B, 1110 B, 0000 B,
3|0100 B, 0100 B, 0100 B, 0000 B,
4|0020 B, 1110 B, 2000 B, 0000 B,
5|0002 B, 1112 B, 0000 B, 0000 B,
6|0000 B, 1110 B, 0000 B, 0000 B,
7|0000 B, 0100 B, 0000 B, 0000 B,
8|0000 B, 3000 B, 0000 B, 0000 B,
9|0000 B, 3300 B, 0000 B, 0000 B,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      178-----
0|( MISSIONS- PLAYER'S SHIP EXPLOSION, 1G, SHOOTING SOUND, 1D )
1|HEX DATA IDSCORE ASM
2| A0 87 15 TONES 50 1 -2 2 MOVENOISE 1 2 0 MOVESOUND
3| 4A MASTER 1 4 6A 8 RAMBLE 3 COUNTLIMITS
4| 89 ABVOLS 19 MCVOLS PLAY QUIET
5|: 1D IDSCORE BMUSIC ;
6|DATA 1GSCORE ASM
7| 2 1 20 MOVESOUND 53 66 75 TONES 1F MCVOLS 0FF ABVOLS
8| 4 5 1 01F MOVENOISE 3 MASTER 2A -1 3 2 RAMBLE 2 COUNTLIMITS
9| PLAY FF 1F 08 -1 0 0F MOVEVOLS 1 2 10 MOVESOUND
10| 3 COUNTLIMITS RERAMBLE PLAY QUIET
11|-->
12|
13|
14|
15|

```

```

+-----Block      179-----
0|( MISSIONS- ZPIF & PZIP SOUNDS- ZP,PZ ) HEX
1|DATA PZSCORE ASM
2| #G3 #F3 #CS3 TONES 1 -4 3F MOVESOUND 10 MASTER
3| 1 4 A0 10 RAMBLE 1 COUNTLIMITS
4| 10 1 4 70 MOVENOISE AA ABVOLS 2A MCVOLS 4F VIBS PLAY QUIET
5|DATA ZPSCORE ASM
6| #G3 #F3 #CS3 TONES 1 -1 3F MOVESOUND 0 1 4 30 MOVENOISE
7| 60 MASTER 1 -4 60 30 RAMBLE 1 COUNTLIMITS AA ABVOLS 1A MCVOLS
8| PLAY 2B MASTER 1 -4 2F 5 RAMBLE 1 COUNTLIMITS
9| 1E 1 -4 6 MOVENOISE PLAY QUIET
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      180-----
0|( SPACE MISSIONS BMUSIC BLOCK )
1|: 1G 1GSCORE PMUSIC 8 MS 1GSCORE P2MUSIC ;
2|: PZ PZSCORE BMUSIC ;
3|: ZP ZPSCORE BMUSIC ;
4|DECIMAL -->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      181-----
0|( MISSIONS- TAKE-OFF- TO ) HEX
1|DATA T01SCORE ASM
2| 7 9 10 TONES 1 5 0 MOVESOUND 2 MASTER 3 0A 90 2 RAMP
3| 1 COUNTLIMITS 3 1 3 73 MOVENOISE 1C MCVOLS CC ABVOLS PLAY
4| 90 MASTER 3E 44 11 TONES 4 -1 90 74 RAMBLE 1 COUNTLIMITS
5| 1 1 20 MOVESOUND PLAY 74 MASTER 1 -1 74 2 RAMBLE
6| 6B 1 -1 3 MOVENOISE RERAMBLE 1 COUNTLIMITS PLAY QUIET
7|DATA T02SCORE ASM
8| 3 5 14 TONES 1 -5 3F MOVESOUND 2 MASTER 3 0A 90 2 RAMP
9| 1 COUNTLIMITS 3 1 3 73 MOVENOISE 1C MCVOLS CC ABVOLS PLAY
10| 2 MASTER 54 31 13 TONES 1 1 22 2 RAMBLE 1 COUNTLIMITS
11| 1 -1 20 MOVESOUND PLAY 22 MASTER 1 1 70 22 RAMBLE
12| 6B 1 -2 3 MOVENOISE RERAMBLE 1 COUNTLIMITS PLAY QUIET
13|: TO T02SCORE P2MUSIC T01SCORE PMUSIC ;
14|-->
15|

+-----Block      182-----
0|( MISSIONS- DIVE SOUND ) HEX
1|DATA KBSCORE ASM ( no priority for background )
2| 1 -4 3F MOVESOUND #G0 #B0 #D4 TONES 77 ABVOLS 7 MCVOLS 48 VIBS
3| 2 MASTER 1 1 50 2 RAMBLE 1 COUNTLIMITS PLAY
4| 55 ABVOLS 25 MCVOLS 1 -1 50 30 RAMBLE HERE 0 1 2 3A MOVENOISE
5| 1 2 0 MOVESOUND 1 COUNTPANS PLAY 3A 1 -2 0 MOVENOISE
6| 1 COUNTPANS PLAY LDPCC
7|
8|SUBR PLAYKBS KBSCORE H LXI, MB2 Y LXIX, bmusic JMP,
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

+-----Block      183-----
0|( DRAW FIREBASES ON SCREEN )
1|HEX TABLE FBADDRESSES 200 , 1000 , 1E00 , 2C00 , 3A00 ,
2|: REDRAWFB FBCOUNTER @ IF
3|FBCOUNTER @ 0 DO
4|100 I FBADDRESSES @ SMALBASE 20 WRITEP LOOP THEN ;
5|DECIMAL -->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      184-----
0|( GAME VARIABLES AND CONSTANTS )
1|TIMER0 C= WTIMER TIMER1 C= BOMBTIMER TIMER2 C= ATTACKTIMER
2|TIMER3 C= RACKTIMER 0 V= FIREACTION 0 V= SPIRALRATE
3|0 V= RACKDONE 0 V= INVADERSLEFT 0 V= PLAYERHIT 0 V= ENDOFFRAME
4|0 V= GAMEOVER 0 V= PHASINTR 0 V= FIRETRACK 0 V= ATTACKWAITER
5|0 V= SCRPTTR 0 V= PV1 2 V= MAXBOMBS 4 ARRAY BVLIST
6|0 V= FBANIM 0 V= REINIT ( MUST REINITIALIZE )
7|10 C= NBOMBS 6 C= BOMBASIZE NBOMBS BOMBASIZE * C= BATOTAL
8|BATOTAL BA= BOMBARRAY
9|HEX 40 BA= FBVECTOR
10|0 V= PINTERFLAG 0 V= PINTERN 0 V= PINTERX 0 V= PINTERY
11|10 C= COINSW 10 C= SW1PG 1 C= SWFIRE 9 C= PLYRINTX 0F C= FFXLL
12|0 C= INIVLL 4200 C= INIVUL -200 C= BOMBDX 700 C= BOMBDY
13|200 C= INIDMY
14|DECIMAL
15|-->

+-----Block      185-----
0|( INITIALIZE GAME SCREEN ) HEX
1|: INITSCREEN DI GRAPHICS 1 CONCM OUTP INITFREELIST
2|NILVQ FIREUP CL CB VERBL OUTP 0D HORCB OUTP ;
3|: UPDATESCORE PLAYERUP @ IF 0 P2SCR BCD+! 4900 9600 408 0 P2SCR
4|ELSE 0 P1SCR BCD+! 4900 0 408 0 P1SCR THEN DI DISPBCD6 E! ;
5|: INI1PG 4900 0 408 0 P1SCR DISPBCD6
6|4D00 0B00 408 A" ONE" COUNT SPOST
7|4D00 4800 408 A" MISSION" COUNT SPOST
8|49C0 5A00 428 MISSIONCTR @ DISPBCD2 REDRAWFB ;
9|: INI2PG INI1PG 4900 9600 408 0 P2SCR DISPBCD6
10|4D00 A100 408 A" TWO" COUNT SPOST ;
11|: DRAWMISSIONSCREEN INITSCREEN NPLAYERS @ IF INI2PG ELSE INI1PG
12|THEN START ;
13|DECIMAL -->
14|
15|

+-----Block      186-----
0|( RACK UPDATOR )
1|HEX 0 V= INVADERNUM 0 V= MASTERY 0 V= DMASTERY
2|0 V= MASTERX 0 V= DMASTERX 0 V= INVLL 0 V= INVUL
3|0 V= LEFTINVN 0 V= RIGHTINVN
4|0 V= BUMPMASSTERROUTINE 0 V= NORMLP1 0 V= NORMLP2
5|8 BARRAY ALIVEBITS
6|8 BARRAY RACKBITS DATA BITMASK 1 B, 2 B, 4 B, 8 B, 10 B, 20 B,
7|40 B, 80 B, 80 ARRAY ANIMSTATE
8|8 BARRAY RACKEXPS 0 V= INVSTATD
9|: RESETTRACK 0 INVADERNUM DI 0 MASTERY ! 0 MASTERX !
10|INIDMY DMASTERY !
11|FE00 DMASTERX ! INIVLL INVLL ! INIVUL INVUL ! ;
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      187-----
0|( RADIAL LINE GENERATOR )
1|0 C= LTMR
2|1 C= LDXL
3|2 C= LDXH
4|3 C= LXL
5|4 C= LXH
6|5 C= LDYL
7|6 C= LDYH
8|7 C= LYL
9|8 C= LYH
10|9 C= LXRL
11|10 C= LXRH
12|11 C= LYR
13|12 C= LRTMR
14|13 C= LPXV
15|-->

```

```

+-----Block      188-----
0|( RADIAL EFFECT VARIABLES )
1|14 C= LMSIZ 32 C= LACOUNT
2|LMSIZ LACOUNT * C= LASIZ
3|LASIZ BA= LARRAY
4|0 V= LINIT ( INITIALIZE LINE ROUTINE )
5|0 V= LQUAD ( QUADRANT COUNTER ) 0 V= LRADIAL ( ANGLE )
6|0 V= XB 0 V= YB ( BIASES )
7|0 V= XS 0 V= YS 0 V= XF 0 V= YF ( LINE ENDPOINTS )
8|0 V= SFBX 0 V= SFFX 0 V= SFBY 0 V= SFFY ( SCALE FACTORS )
9|0 V= XLP 0 V= XLM 0 V= YLP 0 V= YLM ( LIMIT FACTORS )
10|0 V= SVCX 0 V= SVCY
11|-->
12|
13|
14|
15|

```

```

+-----Block      189-----
0|( NEAT SUBROUTINES )
1|SUBR SINE A E MOV, 0 D MVI, 0 sin-table H LXI, D DAD,
2|M E MOV, RET,
3|SUBR COSINE A E MOV, 63 A MVI, E SUB, A E MOV, 0 D MVI,
4|0 sin-table H LXI, D DAD, M E MOV, RET,
5|( UNSIGNED MULTIPLY )
6|F= MNOSW F= MMPL F= NOADD
7|SUBR UMPY <ASSEMBLE H A MOV, A ANA, MNOSW JRZ, XCHG,
8|LABEL MNOSW L A MOV, 0 H LXI,
9|LABEL MMPL RAR, NOADD JRNC, D DAD,
10|LABEL NOADD XCHG, H DAD, XCHG, A ANA, MMPL JRNZ, RET,
11|ASSEMBLE>
12|SUBR COMHL H A MOV, CMA, A H MOV, L A MOV, CMA, A L MOV,
13|H INX, RET,
14|-->
15|

```

```

+-----Block      190-----
0|( SUBR TO WRITE NEXT PIXEL IN A LINE )
1|HEX F= TMRZ F= RTZ
2|SUBR STEPLINE <ASSEMBLE
3|LTMR X A LDX, A ANA, TMRZ JRZ, A DCR, A LTMR X STX,
4|LDXL X E LDX, LDXH X D LDX, LXL X L LDX, LXH X H LDX, D DAD,
5|L LXL X STX, H LXH X STX, XCHG,
6|LDYL X C LDX, LDYH X B LDX, LYL X L LDX, LYH X H LDX, B DAD,
7|L LYL X STX, H LYH X STX,
8|20 C MVI, relabs CALL, C A MOV, DI, MAGIC OUT, 0C0 M MVI, EI,
9|RET, LABEL TMRZ LRTMR X A LDX, A ANA, RTZ JRZ,
10|A LTMR X STX, 0 LRTMR X MVIX,
11|LXRL X L LDX, LXRH X H LDX, L LXL X STX, H LXH X STX,
12|LYR X A LDX, A LYH X STX, 0 LYL X MVIX, RET,
13|LABEL RTZ LINIT LHLD, PCHL,
14|ASSEMBLE>
15|DECIMAL -->

```

```

+-----Block      191-----
0|( OTHER NEAT VERBS )
1|: LSTART LINIT ! LASIZ 0 DO 0 I LARRAY B! LOOP ;
2|F= UPAL
3|CODE UPDATEALL <ASSEMBLE X PUSHX, Y PUSHX, EXX, LACOUNT B LXI,
4|0 LARRAY X LXIX,
5|LABEL UPAL B PUSH, STEPLINE CALL,
6|LMSIZ D LXI, D DADX,
7|B POP, B DCX, C A MOV, B ORA, UPAL JRNZ,
8|EXX, Y POPX, X POPX, NEXT ASSEMBLE>
9|-->

```

```

10|
11|
12|
13|
14|
15|

```

```

+-----Block      192-----
0|( GENERATE A LINE )
1|HEX F= DOY F= OKX
2|SUBR GENLINE <ASSEMBLE random CALL, L A MOV, 3 ANI, LQUAD STA,
3|D A MOV, 3F ANI, LRADIAL STA,
4|( X START )
5|SINE CALL, SFBX LHLD, UMPY CALL, H A MOV, XS STA,
6|( X END )
7|LRADIAL LDA, SINE CALL, SFFX LHLD, UMPY CALL,
8|H A MOV, XF STA,
9|( Y START )
10|LRADIAL LDA, COSINE CALL, SFBY LHLD, UMPY CALL,
11|H A MOV, YS STA,
12|( Y END )
13|LRADIAL LDA, COSINE CALL, SFFY LHLD, UMPY CALL,
14|H A MOV, YF STA,
15|-->

```

```

+-----Block      193-----
0|( LINE GENERATOR - CLIP CHECK )
1|LQUAD LDA, 2 ANI, 0=, IF, XLM LDA, ELSE, XLP LDA, THEN,
2|A C MOV, XF LDA, C CMP, DOY JRC, ( JUMP IF OK )
3|B PUSH, YS LDED, YF LDA, E SUB, A E MOV,
4|XF LDA, C SUB, A L MOV, 0 H MVI, H D MOV, UMPY CALL,
5|XS LDED, XF LDA, E SUB, A E MOV, 0 D MVI, UNSDIV CALL,
6|YF LDA, E SUB, YF STA, B POP, C A MOV, XF STA,
7|( Y STUFF )
8|LABEL DOY LQUAD LDA, A INR, 3 ANI, 2 CPI, CY, IF, YLM LDA,
9|ELSE, YLP LDA, THEN, A C MOV, YF LDA, C CMP, OKX JRC,
10|B PUSH, XS LDED, XF LDA, E SUB, A E MOV,
11|YF LDA, C SUB, A L MOV, 0 H MVI, UMPY CALL,
12|YS LDED, YF LDA, E SUB, A E MOV, 0 D MVI, UNSDIV CALL,
13|XF LDA, E SUB, XF STA, B POP, C A MOV, YF STA,
14|-->
15|

+-----Block      194-----
0|( LINE GENERATOR - SET DELTAS )
1|LABEL OKX XS LDED, XF LDA, E SUB, A C MOV, YS LDED, YF LDA,
2|E SUB, A B MOV, C CMP, CY, IF, ( X IS LARGER )
3|C LRTMR X STX, C LTMR X STX, 40 LDXL X MVIX, 0 LDXH X MVIX,
4|B H MOV, 0 L MVI, C E MOV, L D MOV, UNSDIV CALL,
5|E LDYL X STX, D LDYH X STX,
6|ELSE, ( Y IS LARGER )
7|B LRTMR X STX, B LTMR X STX, 0 LDYL X MVIX,
8|1 LDYH X MVIX, C A MOV, RRC, RRC, A H MOV, 0C0 ANI, A L MOV,
9|H A MOV, 3F ANI, A H MOV, B E MOV, 0 D MVI, UNSDIV CALL,
10|E LDXL X STX, D LDXH X STX,
11|THEN,
12|-->
13|
14|
15|

+-----Block      195-----
0|( ADJUST DELTAS TO QUADRANT, AND BIAS TO EFFECT CENTER )
1|XS LDA, RRC, RRC, A D MOV, 0C0 ANI, A E MOV, D A MOV,
2|3F ANI, A D MOV, XB LHLD,
3|LQUAD LDA,
4|2 ANI, 0<>, IF, A ANA, D DSBC, XCHG, LDXL X L LDX,
5|LDXH X H LDX, COMHL CALL, L LDXL X STX, H LDXH X STX,
6|XCHG, ELSE, D DAD,
7|THEN, L LXL X STX, H LXH X STX, L LXRL X STX, H LXRH X STX,
8|YS LDA, A D MOV, 0 E MVI, YB LHLD,
9|LQUAD LDA, A INR, 3 ANI, 2 CPI, CY~, IF, A ANA, D DSBC,
10|XCHG, LDYL X L LDX, LDYH X H LDX, COMHL CALL,
11|L LDYL X STX, H LDYH X STX, XCHG,
12|ELSE, D DAD, THEN,
13|0 LYL X MVIX, H LYH X STX, H LYR X STX,
14|RET, ASSEMBLE> .NOFF
15|DECIMAL -->

```



```

+-----Block      196-----
0|( WRITE ONLY ENTRY AND SET CENTER OF LINE EFFECT )
1|
2|SUBR WRTONLY GENLINE CALL, 0 LRTMR X MVIX, RET,
3|
4|: SETLXY 2DUP YB ! XB !
5|256 / DUP YLP ! 192 SWAP - YLM !
6|64 / DUP DUP 255 > IF DROP 255 THEN XLP !
7|292 SWAP - DUP 255 > IF DROP 255 THEN XLM ! ;
8|: SETSF SFFY ! SFFX ! SFBY ! SFBX ! ;
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      197-----
0|( SHIFT RIGHT ARITHMETIC BY N ROUTINE )
1|( ENTER AT SRHLC TO CHECK FOR 0 S.A. )
2|F= SRHL F= SAZ
3|SUBR SRHLC <ASSEMBLE B A MOV, A ANA, SAZ JRZ,
4|LABEL SRHL H SRAR, L RARR, SRHL DJNZ, RET,
5|( ZERO OUT HL )
6|LABEL SAZ A L MOV, A H MOV, RET, ASSEMBLE>
7|-->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      198-----
0|( SPIRAL VECTOR ROUTINE )
1|HEX F= SOF SUBR VSPIRAL <ASSEMBLE
2|VDYL X L LDX, VDYH X H LDX, VDDXL X B LDX, SRHL CALL,
3|XCHG, VDXL X L LDX, VDXH X H LDX, A ANA, D DSBC,
4|L E MOV, H D MOV, VDDXH X B LDX, SRHLC CALL, D DAD,
5|L VDXL X STX, H VDXH X STX, SVCX LBCD, B DAD,
6|H A MOV, 50 CPI, SOF JRNC, L VXL X STX, H VXH X STX, XCHG,
7|VDDYL X B LDX, SRHL CALL,
8|VDYL X E LDX, VDYH X D LDX, D DAD,
9|L E MOV, H D MOV, VDDYH X B LDX, SRHLC CALL,
10|D DAD, L VDYL X STX, H VDYH X STX,
11|7 VIDENT X BITX, 0<>, IF, COMHL CALL, THEN,
12|SVCY LBCD, B DAD, H A MOV, 000 CPI, SOF JRNC,
13|L VYL X STX, H VYH X STX, RET,
14|LABEL SOF PQSRH PQS X RESX, PQSDW PQS X SETX, RET, ASSEMBLE>
15|DECIMAL -->

```

```

+-----Block      199-----
0|(< INTERRUPT ROUTINE TO SPIRAL VECTOR )
1|F= SPIL
2|SUBR SPWRITE <ASSEMBLE TBCALC CALL, SPIRALRATE LDA,
3|LABEL SPIL PSW PUSH, VSPIRAL CALL, PSW POP, A DCR, SPIL JRNZ,
4|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
5|PQSDE PQS X RESX, THEN,
6|aup CALL,
7|PQSDW PQS X BITX, 0=, IF, VIWRITE CALL, THEN,
8|KILLOFF JMP, ASSEMBLE>
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      200-----
0|(< SUBROUTINES TO CALCULATE DISPLACEMENTS FOR RACK MEMBER ) HEX
1|SUBR CALCIN VX 7 ANI, RLC, RLC, A H MOV, 0 L MVI, RET,
2|SUBR CALCIN VY 38 ANI, RLC, A H MOV, 0 L MVI, RET,
3|(< CHECK FOR SCREEN EDGE, NEGATE DELTA AND BUMP X IF AT IT )
4|SUBR FLIPCHECK H A MOV, B CMP, RNZ, L A MOV, C CMP, RNZ,
5|D A MOV, CMA, A D MOV, E A MOV, CMA, A E MOV, D INX, A XRA,
6|RET,
7|(< INDEX RACK BITS AND ALIVE BITS )
8|SUBR XRACKBITS C A MOV, 7 ANI, A E MOV, 0 D MVI,
9|BITMASK H LXI, D DAD, M B MOV, C A MOV, RRC, RRC,
10|RRC, 7 ANI, A E MOV, 0 RACKBITS H LXI, D DAD,
11|M A MOV, B ANA, RET,
12|SUBR XALIVEBITS C A MOV, 7 ANI, A E MOV, 0 D MVI,
13|BITMASK H LXI, D DAD, M B MOV, C A MOV, RRC, RRC, RRC, 7 ANI,
14|A E MOV, 0 ALIVEBITS H LXI, D DAD, M A MOV, B ANA, RET,
15|-->
+-----Block      201-----
0|(< WAIT AND ANIMATION TRACKING TABLE ROUTINES ) HEX
1|: WAIT WTIMER ! BEGIN WTIMER @ 0= END ;
2|SUBR NOTEANIM INVADERNUM LDA, RLC,
3|RLC, A E MOV, 0 D MVI, 0 ANIMSTATE H LXI, D DAD, XCHG,
4|MASTERY LHLD, DMASTERY LBCD, 7 B BIT, 0=, IF, B DAD, THEN,
5|XCHG, DI, E M MOV, H INX, D M MOV, H INX, MASTERX LDOD,
6|E M MOV, H INX, D M MOV, EI, RET,
7|SUBR GETASTATE C A MOV, RLC, RLC, A E MOV, 0 D MVI,
8|0 ANIMSTATE H LXI, D DAD, M E MOV, H INX, M D MOV, H INX,
9|H PUSH, C A MOV, CALCIN VY CALL, D DAD, D POP, H PUSH, D PUSH,
10|H BIT, NORMLP1 LHLD, 0<>, IF, 10 D LXI, D DAD, THEN,
11|C A MOV, 7 ANI, RLC, A E MOV, 0 D MVI, D DAD,
12|M E MOV, H INX, M D MOV, D PUSH, Y POPX,
13|C A MOV, H POP, M E MOV, H INX, M D MOV, CALCIN VX CALL, D DAD,
14|XCHG, H POP, RET,
15|-->

```

```

+-----Block      202-----
0|( RECOMPUTE LIMITS ) HEX
1|F= LLFL F= LLFND F= ULFL F= ULFND
2|SUBR RELMT <ASSEMBLE INVADERSLEFT LDA, A ANA,
3|RZ, INIVLL H LXI, 0 ALIVEBITS D LXI, 800 B LXI,
4|LABEL LLFL D LDAX, A ANA, LLFND JRNZ, H A MOV, 10 SUI,
5|A H MOV, D INX, C A MOV, 8 ADI, A C MOV, LLFL DJNZ, RET,
6|LABEL LLFND INVLL SHLD, 7 ALIVEBITS D LXI, INIVUL H LXI,
7|C A MOV, LEFTINVN STA, 38 C MVI,
8|LABEL ULFL D LDAX, A ANA, ULFND JRNZ, H A MOV, 10 ADI,
9|A H MOV, D DCX, C A MOV, 8 SUI, A C MOV, ULFL JMPR,
10|LABEL ULFND INVUL SHLD, C A MOV, RIGHTINVN STA, RET, ASSEMBLE>
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      203-----
0|( SUBR TO STEP MASTER COORDS ONE TICK AND LIMIT CHECK ) HEX
1|( ROUTINE TO WRITE ONE INVADER, IF POSSIBLE )
2|F= INVFINF F= STEPMASER F= NONERD
3|SUBR WRITEINVADER <ASSEMBLE
4|INVADERNUM LDA, A C MOV, XALIVEBITS CALL, NONERD JRZ,
5|DI, XRACKBITS CALL, INVFINF JNZ, NOTEANIM CALL, EI,
6|LABEL NONERD INVADERNUM LDA, A INR, 3F ANI, INVADERNUM STA,
7|WRITEINVADER JRNZ,
8|LABEL STEPMASER BUMPMASERROUTINE LHLD, PCHL,
9|-->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      204-----
0|( WE FOUND AN INVADER - WRITE HIM )
1|LABEL INVFINF INVADERNUM LDA, CALCINVT CALL, MASTERY LDED,
2|D DAD, XCHG, INVADERNUM LDA, 7 ANI, RLC, A C MOV,
3|0 B MVI, INVPATAB LHLD, B DAD, M C MOV, H INX, M B MOV,
4|B PUSH, XTIY, 20 B LXI, 1 D BIT, 0<>, IF, 1 Y H LDX, H DCR,
5|0 L MVI, D DAD, XCHG, A0 C MVI, THEN, B PUSH,
6|INVADERNUM LDA,
7|CALCINVT CALL, MASTERX LBCD, B DAD, XCHG, B POP, relabs CALL,
8|DI, writep CALL, NOTEANIM CALL, Y POPX, INVADERNUM LDA,
9|A INR, 3F ANI, INVADERNUM STA, STEPMASER JZ, RET,
10|ASSEMBLE>
11|CODE WRTINV RACKTIMER LDA, A ANA, 0=, IF, B PUSH, X PUSHX,
12|WRITEINVADER CALL, X POPX, B POP, THEN, NEXT
13|DECIMAL -->
14|
15|

```

```

+-----Block      205-----
0|( REWRITE A RACK MEMBER USING NORMAL PATTERNS )
1|HEX
2|( USED FOR GAME INITIALIZATION )
3|F= TOGGLEMEMBER
4|SUBR REWRITEMEMBER <ASSEMBLE XRACKBITS CALL, RZ,
5|LABEL TOGGLEMEMBER
6|GETASTATE CALL,
7|20 B LXI, relabs CALL, DI, writep CALL, EI, RET,
8|ASSEMBLE>
9|CODE REWRITER H POP, B PUSH, Y PUSHX, L C MOV,
10|REWRITEMEMBER CALL, Y POPX, B POP, NEXT
11|DECIMAL -->
12|
13|
14|
15|

+-----Block      206-----
0|( REENTER RACK ) HEX
1|SUBR PLOTREENTRY
2|VRACK X C LDX, GETASTATE CALL,
3|0 B MVI, H VYH X STX,
4|28 VXZW X MVIX,
5|XCHG, VXL X A LDX, 0C0 ANI, A E MOV,
6|VXH X D LDX, A ANA, D DSBC,
7|0<>, IF, H A MOV, A ANA, 0<, IF, -40 H LXI, ELSE, 40 H LXI,
8|THEN, D DAD, L VXL X STX, H VXH X STX, B INR, THEN,
9|B A MOV, A ANA, 0=, IF,
10|Y PUSHX, H POP, L VPATL X STX, H VPATH X STX,
11|PQSRH PQS X RESX, VRACK X C LDX, XRACKBITS CALL,
12|B A MOV, M ORA, A M MOV, 20 VMAGIC X MVIX,
13|THEN, RET,
14|-->
15|

+-----Block      207-----
0|( INTERRUPT ROUTINE TO REENTER A GALAXIAN ) DECIMAL
1|F= ROGER
2|SUBR RENTGAL <ASSEMBLE
3|TBCALC CALL, B PUSH,
4|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE, PQSDE PQS X RESX,
5|THEN, aup CALL, B POP,
6|PQSRH PQS X BITX, 0<>, IF,
7|LABEL ROGER B PUSH, PLOTREENTRY CALL, B POP,
8|PQSRH PQS X BITX, 0<>, IF, C DCR, ROGER JRNZ, THEN,
9|THEN, PQSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE,
10|PQSRH PQS X BITX, 0<>, IF, PQSDW PQS X RESX,
11|PQSDE PQS X SETX, THEN, THEN,
12|KILLOFF JMP, ASSEMBLE>
13|-->
14|
15|

```

```

+-----Block      208-----
0|( CHECK FOR INTERCEPT WITH RACK MEMBER )
1|( RETURNS NZ, C=INVADERNUM IF DETECTED, ELSE Z )
2|F= NORKI
3|HEX SUBR RACKCHECK <ASSEMBLE
4|VXL X L LDX, VXH X H LDX, H INR, H INR,
5|MASTERX LDED, A ANA, D DSBC, H A MOV, A ANA, NORKI JM,
6|RRC, RRC, 7 ANI, A C MOV, VYL X L LDX, VYH X H LDX,
7|MASTERY LDED, A ANA, D DSBC, H A MOV, 2 ADI, RRC,
8|38 ANI, C ORA, A C MOV, XRACKBITS CALL, RET,
9|LABEL NORKI A XRA, RET,
10|ASSEMBLE>
11|DECIMAL -->
12|
13|
14|
15|

+-----Block      209-----
0|( ANIMATION LIST AND ROUTINE TO EXPLODE THE FIREBASE )
1|HEX DATA FBEXPSUB ASM 0 0 SETDC
2|4 6 DISPL FBEXP1 SETP 08 SWAIT A0 SETM 08 SWAIT
3|20 SETM -4 -6 DISPL FBEXP4 SETP 08 SWAIT A0 SETM 08 SWAIT
4|20 SETM 1 2 DISPL FBEXP2 SETP 08 SWAIT A0 SETM 08 SWAIT
5|20 SETM -1 -2 DISPL FBEXP3 SETP 8 SWAIT A0 SETM 8 SWAIT 20 SETM
6|ARET
7|DATA FBEXP ASM FBEXPSUB ACALL FBEXP5 SETP 8 SWAIT A0 SETM 8
8|SWAIT 20 SETM FBEXP6 SETP 8 SWAIT A0 SETM 8 SWAIT NULPAT SETP
9|1 SWAIT AHALT
10|( ROUTINE TO EFFECT THE EXPLOSION )
11|SUBR EXPLODEFB PLAYERHIT LDA, A ANA, 0=, IF,
12|1 A MVI, PLAYERHIT STA, FBEXP H LXI,
13|CRASHA CALL, XAWRITE H LXI, L PQRH X STX, H PQRH X STX, THEN,
14|RET, DECIMAL -->
15|

+-----Block      210-----
0|( SCORIN ) HEX TABLE ASTBL 60 , 60 , 80 , 100 , 300 , 200 ,
1|150 , 250 ,
2|DECIMAL DATA EXPISUB ASM EXPLOSION1 SETP 5 SWAIT EXPLOSION2
3|SETP 5 SWAIT EXPLOSION3 SETP 5 SWAIT EXPLOSION4 SETP 5 SWAIT
4|EXPLOSION5 SETP 5 SWAIT NULPAT SETP ARET
5|DATA EXPINV ASM EXPISUB ACALL AHALT
6|DATA EXPNS ASM EXPISUB ACALL NUMWRITE SETR 1 SWAIT ARET
7|DATA EXPNF ASM 1 SWAIT HEX BF 40 SETS 1C SWAIT BF 0 SETS AHALT
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      211-----
0|( MORE SCORING GOODIES )
1|DATA SCR60 ASM EXPNS ACALL 0 ASTBL SETP EXPNF AJMP
2|DATA SCR80 ASM EXPNS ACALL 2 ASTBL SETP EXPNF AJMP
3|DATA SCR100 ASM EXPNS ACALL 3 ASTBL SETP EXPNF AJMP
4|DATA SCR300 ASM EXPNS ACALL 4 ASTBL SETP EXPNF AJMP
5|DATA SCR200 ASM EXPNS ACALL 5 ASTBL SETP EXPNF AJMP
6|DATA SCR150 ASM EXPNS ACALL 6 ASTBL SETP EXPNF AJMP
7|HEX DATA EXPTHEGORF ASM FBEXPSUB ACALL FBEXP5 SETP 40 SWAIT
8|A0 SETM 40 SWAIT 20 SETM FBEXP6 SETP 40 SWAIT A0 SETM 40 SWAIT
9|20 SETM NULPAT SETP NUMWRITE SETR 1 SWAIT 7 ASTBL SETP
10|EXPNF AJMP
11|DATA ATTACKEXPTBL SCR60 , SCR60 , SCR80 , SCR100 , SCR300 ,
12|SCR200 , SCR150 , EXPTHEGORF , DECIMAL -->
13|
14|
15|

```

```

+-----Block      212-----
0|( BACKGROUND PHASOR INTERCEPT PROCESSING ROUTINES )
1|( ROUTINE TO EXPLODE AN INVADER )
2|HEX SUBR PINTERPROC A DCR, 0=, IF, PINTERN LDA, 7 ANI, RLC,
3|A C MOV, 0 B MVI, ATTACKEXPTBL H LXI, B DAD, M C MOV, H INX,
4|M B MOV, PINTERX LDED, PINTERY LHLD,
5|ELSE, PINTERN LDA, A C MOV,
6|DI, XRACKBITS CALL, M XRA, A M MOV, XALIVEBITS CALL, M XRA,
7|A M MOV, EI, B PUSH, TOGGLEMEMBER CALL, B POP, GETASTATE CALL,
8|EXPINV B LXI, THEN, D PUSH, H PUSH, B PUSH,
9|PINTERN LBCD, B PUSH, 6 C BIT, 0=, IF, INVADERSLEFT H LXI,
10|M DCR, THEN, 0A2 B LXI, B PUSH, XYVSTART JMP,
11|( ROUTINE TO CHECK FOR INTERCEPT )
12|SUBR PINTERCHECK PINTERFLAG LDA, A ANA, RZ, PINTERPROC CALL,
13|RELMT CALL,
14|PINTERFLAG LDED, PINTERN LHLD, A XRA, PINTERFLAG STA, A INR,
15|RET, DECIMAL -->

```

```

+-----Block      213-----
0|( ROUTINE TO CALL FROM SCAN LOOP )
1|CODE PIFCHECK X PUSHX, Y PUSHX, EXX, PINTERCHECK CALL,
2|Y POPX, X POPX, 0<>, IF, H PUSH, D PUSH, 1 H LXI, ELSE,
3|0 H LXI, THEN, H PUSH, EXX, NEXT
4|HEX : PHASORINTERCEPTCHECK PIFCHECK IF 1 = IF 7 AND ASTBL @ ZP
5|ELSE DROP 50 PZ THEN
6|UPDATESCORE THEN ;
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      214-----
0|( ANIMATION SUBR TO INITIALIZE THE FIRE BASE )
1|( NEEDS INTERCEPT AND LIMITS SET BEFORE CALL )
2|HEX DATA PLAYERANIM ASM JOYWRITE SETR 5700 SETYC 0600 SETXC
3|20 100 SETDC
4|FOREVER FIREBASE SETP 78 SWAIT EVERFOR
5|( ROUTINE TO ACTIVATE THE FIREBASE )
6|: ACTFB FBANIM @ 0 0B2 0 FBVECTOR XVSTART ;
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      215-----
0|( EXPLODE THE FINAL FIREBASE SOMEWHAT MORE SPECTACULAR )
1|HEX
2|( 0D, 13 ARE VXL AND VYL )
3|: KILLLAST WRONLY LINIT ! 0D FBVECTOR @ 200 +
4|13 FBVECTOR @ 600 + SETLXY
5|2 2 28 28 SETSF DI UPDATEALL EI
6|0C0 0 DO UPDATEALL BMS LOOP ;
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      216-----
0|( CHECK FOR PLAYER HIT )
1|HEX
2|: PWAIT WTIMER ! BEGIN BMS PHASORINTERCEPTCHECK WTIMER @ 0=
3|END ;
4|: PLAYERHITCHECK PLAYERHIT @ IF 1G
5|FBCOUNTER @ 0= IF PHASORINTERCEPTCHECK
6|KILLLAST SHUTUP
7|1 GAMEOVER ! 1 ENDOFFRAME !
8|ELSE 97 PWAIT REINIT @ DOIT FBCOUNTER @ 1 - DUP
9|FBCOUNTER !
10|DI FBADDRESSES @ 100 SWAP SMALBASE 20 WRITEP
11|PLAYERHIT ZERO EI
12|INVADERSLEFT @ IF 40 ATTACKTIMER ! ACTFB ELSE 1 ENDOFFRAME !
13|THEN THEN ELSE INVADERSLEFT @ 0= IF TO 30 PWAIT 1 ENDOFFRAME !
14|THEN THEN ;
15|DECIMAL -->

```

```

+-----Block      217-----
0| ( COMMON INITIALIZATION GOODIES )
1| CODE NULCODE NEXT
2| : REPAINTRACK 64 0 DO I REWRITER LOOP ;
3| HEX
4| : INITMISSIONRAM XDI GRAPHICS
5| 0 1STCLRADDR CLRSize FILL TIMER0 ZERO TIMER1 ZERO TIMER2 ZERO
6| TIMER3 ZERO UNPRIOR 1 MUSICFLAG B!
7| ' SHUTUP REINIT ! NULRET FIREACTION ! ;
8| : STARTGAME 0 P1SCR ZERO 1 P1SCR ZERO 0 P2SCR ZERO 1 P2SCR ZERO
9| 2800 MUTHAX ! 6400 MUTHAY ! 1 MISSIONCTR !
10| 1 FBCOUNTER ! PLAYERUP ZERO NPLAYERS ZERO SHUTUP
11| SKILLFACTOR ZERO ;
12| DECIMAL -->
13|
14|
15|

```

```

+-----Block      218-----
0| ( SPECIAL ROUTINE TO MOVE PHASOR BLAST )
1| ( SUBROUTINE TO XAWRITE WITH INTERCEPT CHECKING )
2| HEX F= PHL
3| SUBR PHWRITE <ASSEMBLE PQT B X C LDX, 0 PQT B X MVIX,
4| LABEL PHL B PUSH, VXH X A LDX, A INR, A VXH X STX,
5| 48 CPI, CY~, IF, PQSRH PQS X RESX, THEN,
6| PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
7| PQSDE PQS X RESX, THEN, PQSRH PQS X BITX, 0<>, IF,
8| VIWRITE CALL, THEN,
9| B POP, PQSRH PQS X BITX, 0<>, IF, C DCR, PHL JRNZ,
10| THEN, KILLOFF JMP,
11| ASSEMBLE>
12| DECIMAL -->
13|
14|
15|

```

```

+-----Block      219-----
0| ( START OR RESTART THE PHASOR MOVING )
1| SUBR DOFIREACT C M MOV, FIREACTION LHLD, PCHL,
2| HEX SUBR SETPXY
3| 0 FBVECTOR Y LXIX, VXL Y L LDX, VXH Y H LDX, 500 D LXI,
4| D DAD, L VXL X STX, H VXH X STX, VYL Y L LDX, VYH Y H LDX,
5| 700 D LXI, D DAD, L VYL X STX, H VYH X STX, 60 VATMR X MVIX,
6| RET,
7| SUBR SHOOTPHASOR DOFIREACT CALL,
8| CLRVEC CALL, SETPXY CALL, 0B6 PQS X MVIX,
9| 20 VMAGIC X MVIX, 30 VXZW X MVIX, PHWRITE H LXI,
10| L PQL X STX, H PQRH X STX, PHASINTR LHLD, L VIRL X STX,
11| H VIRH X STX,
12| PBURST H LXI, L VPATL X STX, H VPATH X STX, 1 VDXH X MVIX,
13| STARTVEC CALL, RET,
14| SUBR RESHOOTPHASOR DOFIREACT CALL, SETPXY CALL, RET,
15| DECIMAL -->

```


+-----Block 220-----

```
0|( CHECK FIRE SWITCH )
1|F= FIREBUT
2|SUBR FIRESWCK <ASSEMBLE
3|FIRETRACK H LXI, JOYSTICK IN, A C MOV, M XRA, SWFIRE ANI,
4|RZ, C A MOV, SWFIRE ANI, FIREBUT JRZ, C M MOV, RET,
5|LABEL FIREBUT PV1 LIXD, DI, PQSRH PQS X BITX, 0<>, IF,
6|( KICKOUT IF PHASOR EXPLOSION IN PROGRESS )
7|PQSFRZ PQS X BITX, RNZ, RESHOOTPHASOR CALL,
8|ELSE, PQS X A LDX, A ANA, RNZ, SHOOTPHASOR CALL, THEN,
9|EI, IDSCORE H LXI, 0 MUSIC-BARRAY-1 Y LXIX, bmusic JMP,
10|ASSEMBLE>
11|HEX SUBR FSLITE A XRA, PV1 LIXD, PQSRH PQS X BITX, 0=, IF,
12|A INR, THEN, 26 OUT, RET,
13|CODE FIRECHECK X PUSHX, Y PUSHX, EXX, FSLITE CALL,
14|FIRESWCK CALL, EI, EXX, Y POPX, X POPX, NEXT
15|DECIMAL -->
```

+-----Block 221-----

```
0|( AWAIT THE ARRIVAL OF THE VERTICAL INTERVAL )
1|HEX
2|F= WVIL
3|CODE WVI <ASSEMBLE
4|DI, 11 A MVI, INMOD OUT,
5|VERAF IN, A E MOV,
6|LABEL WVIL VERAf IN, E CMP, WVIL JZ, 0D0 CPI, WVIL JC,
7|0E0 CPI, WVIL JNC,
8|8 A MVI, INMOD OUT,
9|NEXT ASSEMBLE>
10|DECIMAL -->
```

```
11|
12|
13|
14|
15|
```

+-----Block 222-----

```
0|( NEW COLOR ROUTINES )
1|HEX 8 BA= COLTBL 0 V= TARGETCT
2|CODE MAKECOLS EXX, B POP, 0 COLTBL H LXI, TARGETCT LDED,
3|8 B MVI, BEGIN, D LDAX, F8 ANI, C ORA, A M MOV, H INX,
4|D INX, LOOP, EXX, NEXT
5|CODE APPROACHL EXX, B POP, 0 COLTBL H LXI, 0 E MVI, 8 B MVI,
6|BEGIN, M A MOV, 7 ANI, C CMP, 0<>, IF,
7|C A MOV, A ANA, 0=, IF, M DCR, ELSE, M INR, THEN, E INR,
8|THEN, H INX, LOOP, 0 H LXI, E A MOV, A ANA, 0=, IF, H INX,
9|THEN, H PUSH, EXX, NEXT
10|CODE APPROACHC EXX, 0 COLTBL H LXI, TARGETCT LDED, 800 B LXI,
11|BEGIN, D LDAX, M CMP, 0<>, IF, CY, IF, M DCR, ELSE, M INR,
12|THEN, C INR, THEN, H INX, D INX, LOOP,
13|0 H LXI, C A MOV, A ANA, 0=, IF, H INX, THEN, H PUSH, EXX, NEXT
14|DECIMAL -->
15|
```

```

+-----Block      223-----
0|( FADE UP/DOWN ROUTINES )
1|0 V= CWTMR
2|: DC 0 COLTBL WVI COLOR EI ;
3|: CWAIT CWTMR @ PWAIT ; : SCT CWTMR ! ;
4|: STC TARGETCT ! ;
5|: SC STC 8 0 DO I TARGETCT @ + B@ I COLTBL B! LOOP DC ;
6|: FUC STC SCT 0 FLOOD 9 STARZ OUTP CWAIT 0 MAKECOLS DC CWAIT
7|BEGIN APPROACHC DC CWAIT END ;
8|: FDB SCT BEGIN 0 APPROACHL DC CWAIT END 0 WVI FLOOD
9|0 STARZ OUTP EI ;
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      224-----
0|( FORCE FIELD DRAWER ) DECIMAL
1|TIMER2 C= FFTIMER 0 V= DDXC
2|192 BARRAY FIELDADR 0 V= DXC 0 V= XC 0 V= FFLAG 0 V= FFBIAS
3|: INITFF DDXC ! 0 DXC ! 25600 XC ! 0 96 DO
4|DXC @ DDXC @ + DUP DXC !
5|XC @ + DUP XC ! 256 / DUP I FIELDADR B! 191 I - FIELDADR B!
6|-1 +LOOP 0 FFLAG ! ;
7|HEX F= FFLOOP SUBR FIELDRAW <ASSEMBLE
8|FFBIAS LHLD, C0 B MVI, H PUSH, 0 FIELDADR H LXI,
9|LABEL FFLOOP M A MOV, A ANA, XTHL, 0<>, IF, A C MOV,
10|3 ANI, 20 ORI, MAGIC OUT, C A MOV, XCHG, RRC, RRC, 3F ANI,
11|A L MOV, 0 H MVI, D DAD, FF M MVI, H INX, 0 M MVI, XCHG, THEN,
12|50 D LXI, D DAD, XTHL, H INX, FFLOOP DJNZ, H POP, RET,
13|ASSEMBLE>
14|CODE DRAWFIELD EXX, DI, FIELDRAW CALL, EI, EXX, NEXT
15|DECIMAL -->

```

```

+-----Block      225-----
0|( MORE FORCE FIELD GOODIES )
1|DECIMAL
2|: DRAWFF FFLAG @ 0 = IF FFTIMER @ 0 = IF 1 DI FFLAG !
3|DRAWFIELD EI THEN THEN ;
4|: ERASEFF FFLAG @ IF 0 DI FFLAG ! DRAWFIELD EI THEN ;
5|DECIMAL -->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      226-----
0|( CHECK FOR INTERCEPT WITH ANY OF THE ATTACKERS )
1|F= CNH
2|SUBR CKATRS <ASSEMBLE PINTERFLAG LDA, A ANA, CNH JRNZ,
3|1 C MVI, CHECKALL CALL, CNH JRZ,
4|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
5|VYL Y L LDX, VYH Y H LDX, PINTERY SHLD,
6|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
7|VRACK Y A LDX, PINTERN STA,
8|6 A BIT, 0=, IF, A C MOV, XALIVEBITS CALL, M XRA,
9|A M MOV, THEN, 1 A MVI, PINTERFLAG STA, A ANA,
10|RET,
11|LABEL CNH A XRA, RET,
12|ASSEMBLE>
13|DECIMAL -->
14|
15|

```

```

+-----Block      227-----
0|( POSITION OBJECT RELATIVE TO FORMATION LEADER )
1|SUBR POSREL VVPL X L LDX, VVPH X H LDX, H PUSH, Y POPX,
2|VFXBL X L LDX, VFXBH X H LDX, VXL Y E LDX, VXH Y D LDX,
3|D DAD, L VXL X STX, H VXH X STX,
4|VFYBL X L LDX, VFYBH X H LDX, VYL Y E LDX, VYH Y D LDX,
5|D DAD, L VYL X STX, H VYH X STX, RET,
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      228-----
0|( INTERRUPT ROUTINE TO WRITE RELATIVE FORMATION MEMBER )
1|SUBR FWRITE TBCALC CALL,
2|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE, PQSDE PQS X RESX,
3|THEN, aup CALL, POSREL CALL,
4|PQSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE, PQSDW PQS X RESX,
5|PQSDE PQS X SETX, THEN, KILLOFF JMP,
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      229-----
0|( LEADER X Y ANIMATION TIME STATUS VECTOR FSTART )
1|CODE FSTART X PUSHX, H POP, Y PUSHX, D POP, EXX,
2|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX,
3|CLRVEC CALL, 6 Y C LDX, C VRACK X STX,
4|14 Y L LDX, 15 Y H LDX, L VFXBL X STX, H VFXBH X STX,
5|VXL D LXI, D DAD, M E MOV, H INX,
6|M D MOV, 12 Y L LDX, 13 Y H LDX, L VFXBL X STX, H VFXBH X STX,
7|D DAD, L VXL X STX, H VXH X STX,
8|14 Y L LDX, 15 Y H LDX, VYL D LXI, D DAD, M E MOV, H INX,
9|M E MOV, 10 Y L LDX, 11 Y H LDX, L VFYBL X STX, H VFYBH X STX,
10|D DAD, L VYL X STX, H VYH X STX,
11|SETSTDW CALL, FWRITE H LXI, L PQRL X STX, H PQRH X STX,
12|ASFLOK VAUXS X SETX,
13|STARTVEC CALL, UNFRAME 14 H LXI, SP DAD, SPHL,
14|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT
15|DECIMAL -->

```

```

+-----Block      230-----
0|( EFFECT REENTRY INTO RACK OR FORMATION )
1|( HL=TARGET X DE=TARGET Y A=TIME BASE )
2|HEX F= TBCD F= YESLOK F= NOTLOK F= STUFX
3|SUBR PLOTRENT <ASSEMBLE D PUSH, L VYL X STX, H VYH X STX,
4|VDXL X C LDX, VDXH X B LDX, 0 H LXI,
5|LABEL TBCD B DAD, A DCR, TBCD JRNZ,
6|XCHG, VXL X A LDX, 0C0 ANI, A C MOV, VXH X B LDX,
7|A ANA, B DSBC, YESLOK JP, ( IF BELOW TARGET, LOCK TO IT )
8|D DAD, 0<, IF, B H MOV, C L MOV, A ANA, D DSBC,
9|NOTLOK JMPR, THEN, ( IF ABOVE AND CLOSER THAN TBCD, LOCK IN )
10|LABEL YESLOK H POP, A XRA, STUFX JMPR,
11|LABEL NOTLOK D POP, 1 A MVI, A ANA,
12|LABEL STUFX L VXL X STX, H VXH X STX, RET,
13|ASSEMBLE> DECIMAL -->
14|
15|

```

```

+-----Block      231-----
0|( INTERRUPT ROUTINE TO REENTER KAMIKAZE )
1|SUBR REKAMI PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
2|PQSDE PQS X RESX, THEN, TBCALC CALL, C A MOV, ( NOTE! )
3|VFVPL X L LDX, VFVPH X H LDX, H PUSH, Y POPX,
4|VFXBL X L LDX, VFXBH X H LDX, VXL Y C LDX, VXH Y B LDX, B DAD,
5|XCHG, VFYBL X L LDX, VFYBH X H LDX, VYL Y C LDX, VYH Y B LDX,
6|B DAD, PLOTRENT CALL,
7|0=, IF, FWRITE H LXI, L PQRL X STX, H PQRH X STX,
8|ASFLOK VAUXS X SETX, PQSFRZ PQS X RESX, THEN,
9|aup CALL,
10|PQSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE, PQSDW PQS X RESX,
11|PQSDE PQS X SETX, THEN, KILLOFF JMP,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      232-----
0|( ROUTINE TO RETARGET AN ATTACKER )
1|HEX SUBR AABS A ANA, RP, CMA, A INR, RET,
2|( ACTUAL TARGETER )
3|SUBR KTARGET H PUSH, VYH X A LDX, A SRLR, A SRLR, A C MOV,
4|VYH FBVECTOR LDA, A SRLR, A SRLR, C SUB, A SRAR, A SRAR,
5|A E MOV,
6|VDYH X B LDX, B SUB, A C MOV, E A MOV, B XRA, C A MOV,
7|0<, IF, C ADD, THEN,
8|A VDDYL X STX, 7 A BIT, 0 A MVI,
9|0<>, IF, CMA, THEN, A VDDYH X STX,
10|VDDYL X A LDX, AABS CALL, 0E ANI, 6 CPI, CY~, IF, 6 A MVI,
11|THEN, A C MOV, 0 B MVI, VPTBL X L LDX, VPTBH X H LDX,
12|B DAD, M E MOV, H INX, M D MOV, E VPATL X STX,
13|D VPATH X STX, H POP, RET,
14|DECIMAL -->
15|

```

```

+-----Block      233-----
0|( ROUTINE TO FLIP OVER ATTACKER )
1|DATA FLIPOVER ASM
2|HEX A0 SETM DECIMAL 0 PATI 4 SWAIT
3|2 PATI 4 SWAIT
4|4 PATI 4 SWAIT
5|6 PATI 4 SWAIT
6|8 PATI 4 SWAIT
7|HEX 20 SETM DECIMAL
8|6 PATI 4 SWAIT
9|4 PATI 4 SWAIT
10|2 PATI 4 SWAIT
11|0 PATI 8 SWAIT
12|ARET
13|-->
14|
15|

```

```

+-----Block      234-----
0|( LEFT ROLL SEQUENCE )
1|DATA LEFTROLL ASM
2|XADDWRITE SETR
3|-3 -2 SETDDC 64 -128 SETDC 0 PATI 8 SWAIT
4|2 PATI 4 SWAIT
5|4 PATI 4 SWAIT
6|6 PATI 4 SWAIT
7|8 PATI 4 SWAIT
8|-3 4 SETDDC 4 SWAIT
9|HEX A0 SETM DECIMAL 6 PATI 4 SWAIT
10|4 PATI 4 SWAIT
11|2 PATI 4 SWAIT 0 PATI 4 SWAIT HEX 20 SETM DECIMAL
12|8 SWAIT 0 4 SETDDC ARET
13|DATA REENTER ASM 19200 SETXC NULPAT SETP 0 0 SETDC
14|0 0 SETDDC 10 SWAIT RENTGAL SETR 2 SWAIT
15|0 PATI 24 SWAIT FLIPOVER ACALL 120 SWAIT AHALT -->

```

```

+-----Block      235-----
0|( RIGHT ROLL SEQUENCE )
1|DATA RIGHTROLL ASM
2|XADDWRITE SETR HEX A0 SETM DECIMAL
3|-3 2 SETDDC 64 128 SETDC 0 PATI 8 SWAIT
4|2 PATI 4 SWAIT
5|4 PATI 4 SWAIT
6|6 PATI 4 SWAIT
7|8 PATI 4 SWAIT
8|-3 -4 SETDDC 4 SWAIT
9|HEX 20 SETM DECIMAL 6 PATI 4 SWAIT
10|4 PATI 4 SWAIT
11|2 PATI 4 SWAIT 0 PATI 4 SWAIT HEX A0 SETM DECIMAL 8 SWAIT
12|0 -1 SETDDC ARET
13|DECIMAL -->
14|
15|

```

```

+-----Block      236-----
0|( KAMIKAZE ATTACK ANIMATION )
1|DATA KAMIATA ASM
2|6 AREPEAT KTARGET ASMCALL 20 SWAIT ALOOP 60 SWAIT
3|19200 SETXC NULPAT SETP 0 0 SETDC 0 0 SETDDC 1 SWAIT
4|REKAMI SETR 64 0 SETDC 10 SWAIT FLIPOVER ACALL
5|FOREVER 0 PATI 120 SWAIT EVERFOR
6|DATA KAMIATL ASM LEFTROLL ACALL KAMIATA AJMP
7|DATA KAMIATR ASM RIGHTROLL ACALL KAMIATA AJMP
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      237-----
0|( ANIMATION TO ACTIVATE KAMIKAZES )
1|DECIMAL
2|DATA KAMITBL KAMI , KMKZ1R , KMKZ2R , KMKZ3R , KMKZ4R ,
3|DATA AKAMI ASM KAMITBL SETPT 0 PATI FOREVER 120 SWAIT EVERFOR
4|DATA AGORFPT GORF4 , GORF4 , GORF4 , GORF4 , GORF4 ,
5|DATA AKGORF ASM AGORFPT SETPT 0 PATI FOREVER 120 SWAIT EVERFOR
6|DECIMAL ;S
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

100|( SPACE INVADERS GAME )
101|( CRAB1A )
102|( CRAB1B )
103|( CRAB2A )
104|( CRAB2B )
105|( CRAB3A )
106|( CRAB3B )
107|( UFO ) DECIMAL DATA UFO 3 B, 18 B, BINARY 0 B, 0 B, 0 B,
108|( UFO DATA TABLE CONTINUED )
109|( ADDITIONAL UFO PATTERNS )
110|( YET ANOTHER UFO PATTERN )
111|( BOMB PATTERNS FOR SPACE INVADERS )
112|( BOMB PATTERNS STYLE 2 )
113|( MISSIONS- INVADERS THUMP, TH ) HEX
114|( MISSIONS- GORF, IA, GORF EXPLOSION, GE ) HEX
150|( SPACE INVADERS GAME )
151|( MORE GOODIES ) DECIMAL
152|( CONTINUED PATTERN MAKER )
153|( SPACE INVADERS RACK COORDINATE BUMPER ROUTINE )
154|( LOCAL FORCE FIELD GOODIES )
155|( FIREBASE STUFF FOR SPACE INVADERS )
156|( GOODIES TO EXPLODE A BOMB )
157|( CHECK FOR BOMB INTERCEPT ) HEX
158|( BOMB INTERCEPT CHECKER CONTINUED )
159|( PHASOR INTERCEPT CHECK ROUTINE )
160|( BOMB ANIMATION SCORES )
161|( SUBROUTINE TO DROP A BOMB IF POSSIBLE )
162|( BOMBER CONTINUED )
163|( GORF BOUNCE ANIMATION )
164|( UFO1 ANIMATION )
165|( ANIMATION SEQUENCES FOR UFO )
166|( UFO ANIMATION CONTINUED )
167|( SUBROUTINE TO SEND OVER UFOS )
168|( SPACE INVADERS INITIALIZATION ) HEX
169|( CRAB RACK ENTRY ANIMATION )
170|( GORF SPIRAL OUT )
171|( GAME START SPIRAL OUT THE GORF )
172|( SPACE INVADERS DUMPOUT SEQUENCE )
173|( TRANSITION FROM MISSION 1 TO MISSION 2 ) HEX
174|( CRUDE SPACE INVADERS SCAN LOOP )
180|( GORF SPIRAL OUT )
181|( GAME START SPIRAL OUT THE GORF )

```

```

+-----Block      100-----
0|( SPACE INVADERS GAME )
1|DATA GSAB 0 B, 0 ,
2|DECIMAL -->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      101-----
0|( CRAB1A )
1|DECIMAL DATA CRAB1A 3 B, 8 B,
2|BINARY
3|00010001 B, 01000000 B, 0 B,
4|01000101 B, 01010000 B, 0 B,
5|00000001 B, 10010100 B, 0 B,
6|00000101 B, 01010101 B, 0 B,
7|00000101 B, 01010101 B, 0 B,
8|00000001 B, 10010100 B, 0 B,
9|01000101 B, 01010000 B, 0 B,
10|00010001 B, 01000000 B, 0 B,
11|-->
12|
13|
14|
15|

```

```

+-----Block      102-----
0|( CRAB1B )
1|DECIMAL DATA CRAB1B 3 B, 8 B,
2|BINARY
3|01000001 B, 01000000 B, 0 B,
4|00010001 B, 01010000 B, 0 B,
5|01000101 B, 10010100 B, 0 B,
6|00010001 B, 01010101 B, 0 B,
7|00010001 B, 01010101 B, 0 B,
8|01000101 B, 10010100 B, 0 B,
9|00010001 B, 01010000 B, 0 B,
10|01000001 B, 01000000 B, 0 B,
11|-->
12|
13|
14|
15|

```



```

+-----Block      103-----
0|( CRAB2A )
1|DECIMAL DATA CRAB2A 3 B, 11 B, BINARY
2|00000010 B, 10101000 B, 0 B,
3|10001010 B, 10000000 B, 0 B,
4|00101010 B, 10100010 B, 0 B,
5|00001010 B, 01101000 B, 0 B,
6|00001010 B, 10100000 B, 0 B,
7|00001010 B, 10100000 B, 0 B,
8|00001010 B, 10100000 B, 0 B,
9|00001010 B, 01101000 B, 0 B,
10|00101010 B, 10100010 B, 0 B,
11|10001010 B, 10000000 B, 0 B,
12|00000010 B, 10101000 B, 0 B,
13|-->
14|
15|
+-----Block      104-----
0|( CRAB2B )
1|DECIMAL DATA CRAB2B 3 B, 11 B, BINARY 00101010 B, 0 ,
2|00000010 B, 10000000 B, 0 B,
3|00101010 B, 10100010 B, 0 B,
4|10001010 B, 01101000 B, 0 B,
5|10001010 B, 10100000 B, 0 B,
6|00001010 B, 10100000 B, 0 B,
7|10001010 B, 10100000 B, 0 B,
8|10001010 B, 01101000 B, 0 B,
9|00101010 B, 10100010 B, 0 B,
10|00000010 B, 10000000 B, 0 B, 00101010 B, 0 , HEX
11|-->
12|
13|
14|
15|
+-----Block      105-----
0|( CRAB3A )
1|DECIMAL DATA CRAB3A 3 B, 12 B, BINARY
2|11000011 B, 11110000 B, 0 B, 11000011 B, 11111100 B, 0 B,
3|00110011 B, 11111100 B, 0 B, 00111111 B, 00111100 B, 0 B,
4|00001111 B, 00111111 B, 0 B,
5|00110011 B, 11111111 B, 0 B,
6|00110011 B, 11111111 B, 0 B,
7|00001111 B, 00111111 B, 0 B,
8|00111111 B, 00111100 B, 0 B,
9|00110011 B, 11111100 B, 0 B,
10|11000011 B, 11111100 B, 0 B,
11|11000011 B, 11100000 B, 0 B, DECIMAL
12|-->
13|
14|
15|

```

```

+-----Block      106-----
0|( CRAB3B )
1|DECIMAL DATA CRAB3B 3 B, 12 B, BINARY
2|00000011 B, 11110000 B, 0 B, 00110011 B, 11111100 B, 0 B,
3|11111111 B, 11111100 B, 0 B, 11001111 B, 00111100 B, 0 B,
4|00001111 B, 00111111 B, 0 B, 00110011 B, 11111111 B, 0 B,
5|00110011 B, 11111111 B, 0 B, 00001111 B, 00111111 B, 0 B,
6|11001111 B, 00111100 B, 0 B,
7|11111111 B, 11111100 B, 0 B,
8|00110011 B, 11111100 B, 0 B,
9|00000011 B, 11110000 B, 0 B, HEX
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      107-----
0|( UFO ) DECIMAL DATA UFO 3 B, 12 B, BINARY 0 B, 0 B, 0 B,
1|00001100 B, 00000000 B, 0 B,
2|00001111 B, 00000000 B, 0 B,
3|00011111 B, 11000000 B, 0 B,
4|01011101 B, 11110000 B, 0 B,
5|00011111 B, 11110000 B, 0 B,
6|00001111 B, 11111100 B, 0 B,
7|00001101 B, 11111100 B, 0 B,
8|00111111 B, 11111100 B, 0 B,
9|00111111 B, 11111100 B, 0 B,
10|00001101 B, 11111100 B, 0 B,
11|00001111 B, 11111100 B, 0 B,
12|00011111 B, 11110000 B, 0 B,
13|01011101 B, 11110000 B, 0 B,
14|00011111 B, 11000000 B, 0 B,
15|00001111 B, 00000000 B, 0 B,

```

```

+-----Block      108-----
0|( UFO DATA TABLE CONTINUED )
1|00001100 B, 00000000 B, 0 B,
2|0 B, 0 B, 0 B,
3|DECIMAL
4|-->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      109-----
0|( ADDITIONAL UFO PATTERNS )
1|DATA UF02 3 B, 11 B, QUAD
2|0033 B, 0000 B, 0000 B,
3|3033 B, 0000 B, 0000 B,
4|0333 B, 3000 B, 0000 B,
5|0311 B, 3000 B, 0000 B,
6|3333 B, 3300 B, 0000 B,
7|3311 B, 3300 B, 0000 B,
8|3333 B, 3300 B, 0000 B,
9|0311 B, 3000 B, 0000 B,
10|0333 B, 3000 B, 0000 B,
11|3033 B, 0000 B, 0000 B,
12|0033 B, 0000 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      110-----
0|( YET ANOTHER UFO PATTERN )
1|DATA UF03 2 B, 8 B, QUAD
2|0300 B, 0000 B,
3|0320 B, 0000 B,
4|1320 B, 0000 B,
5|0323 B, 0000 B,
6|0323 B, 0000 B,
7|1320 B, 0000 B,
8|0320 B, 0000 B,
9|0300 B, 0000 B,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      111-----
0|( BOMB PATTERNS FOR SPACE INVADERS )
1|QUAD DATA BOMB1 3 B, 3 B, 0010 B, 0010 B, 0 B,
2|0101 B, 0100 B, 0 B, 1000 B, 1000 B, 0 B,
3|DATA BOMB2 3 B, 3 B, 0001 B, 0000 B, 0 B,
4|1010 B, 1010 B, 0 B, 0100 B, 0100 B, 0 B,
5|DATA BOMB3 3 B, 3 B, 0100 B, 0100 B, 0 B,
6|1010 B, 1010 B, 0 B, 0001 B, 0000 B, 0 B,
7|DATA BOMB4 3 B, 3 B, 1000 B, 1000 B, 0 B,
8|0101 B, 0100 B, 0 B, 0010 B, 0010 B, 0 B,
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      112-----
0|( BOMB PATTERNS STYLE 2 )
1|QUAD
2|DATA TOMB1 3 B, 3 B, 0010 B, 0100 B, 0 B, 1111 B, 1110 B, 0 B,
3|0001 B, 0010 B, 0 B,
4|DATA TOMB2 3 B, 3 B, 0100 B, 1000 B, 0 B, 1111 B, 1110 B, 0 B,
5|1001 B, 0000 B, 0 B,
6|DATA TOMB3 3 B, 3 B, 1001 B, 0000 B, 0 B, 1111 B, 1110 B, 0 B,
7|1010 B, 0000 B, 0 B,
8|DATA TOMB4 3 B, 3 B, 1000 B, 0000 B, 0 B, 1111 B, 1110 B, 0 B,
9|1000 B, 0000 B, 0 B,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      113-----
0|( MISSIONS- INVADERS THUMP, TH ) HEX
1|DATA THUMPScore ASM
2|EE 0E 3 -1 0 0E MOVEVOLS 0E HITMO 7 0 0 MOVESOUND
3|#C1 #CS1 #D1 TONES 80 MASTER 36 4 8C 80 RAMP PLAY
4|: TH SHUTUP THUMPScore P2MUSIC ;
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      114-----
0|( MISSIONS- GORF, IA, GORF EXPLOSION, GE ) HEX
1|DATA IAScore ASM
2|3 1 0 MOVESOUND 24 MASTER 1 2 30 20 RAMBLE
3|10 -1 12 MOVELOWLIM 10 -1 2 MOVEHIGHLIM #C2 #E2 #G2 TONES
4|99 ABVOLS 09 MCVOLS PLAY QUIET
5|: IA IAScore P2MUSIC ;
6|DECIMAL ;S
7|DATA GEScore ASM
8|7 9 10 TONES 1 -5 3F MOVESOUND 2 MASTER 1 10 F2 2 RAMP
9|1B COUNTLIMITS 1 1 1 FF MOVENOISE 3C MCVOLS CC ABVOLS PLAY
10|QUIET
11|: GE GEScore P2MUSIC ;
12|-->
13|
14|
15|

```

```

+-----Block      150-----
0|( SPACE INVADERS GAME )
1|TIMER0 C= UFOTIMER
2|-->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      151-----
0|( MORE GOODIES ) DECIMAL
1|4 ARRAY INVADERPAT 4 ARRAY INVADERDROPAT
2|32 BARRAY CRAB1AB 100 BARRAY CRAB1ABD 42 BARRAY CRAB2AB
3|100 BARRAY CRAB2ABD 44 BARRAY CRAB3AB 120 BARRAY CRAB3ABD
4|HEX : MAKEPATS CL 1000 1000 CRAB1A 20 WRITEP 1000 1200 CRAB1B
5|20 WRITEP 6 A 0 CRAB1AB 1000 1000 SNAP 0 CRAB1AB 3 INVADERPAT !
6|1300 2000 CRAB1A 20 WRITEP 1000 2200 CRAB1B 20 WRITEP
7|14 A 0 CRAB1ABD 1000 2000 SNAP 0 CRAB1ABD 3 INVADERDROPAT !
8|2000 1000 CRAB2A 20 WRITEP 2000 1200 CRAB2B 20 WRITEP
9|6 D 0 CRAB2AB 2000 1000 SNAP 0 CRAB2AB 2 INVADERPAT !
10|2300 2000 CRAB2A 20 WRITEP 2000 2200 CRAB2B 20 WRITEP
11|14 D 0 CRAB2ABD 2000 2000 SNAP 0 CRAB2ABD 2 INVADERDROPAT !
12|3000 1000 CRAB3A 20 WRITEP 3000 1200 CRAB3B 20 WRITEP
13|6 E 0 CRAB3AB 3000 1000 SNAP 0 CRAB3AB DUP 0 INVADERPAT !
14|1 INVADERPAT !
15|-->

```

```

+-----Block      152-----
0|( CONTINUED PATTERN MAKER )
1|3300 2000 CRAB3A 20 WRITEP 3000 2200 CRAB3B 20 WRITEP
2|14 E 0 CRAB3ABD 3000 2000 SNAP 0 CRAB3ABD DUP 0 INVADERDROPAT !
3|1 INVADERDROPAT ! ;
4|( SPACE INVADER NORMAL PATTERN TABLE )
5|DATA INVNORMLPAT CRAB3A , CRAB3A , CRAB2A , CRAB1A , 0 , 0 ,
6|0 , 0 , CRAB3B , CRAB3B , CRAB2B , CRAB1B , 0 , 0 , 0 , 0 ,
7|DECIMAL ( MORE GOODIES ) 0 V= UGL
8|: DRGS UGL 1 DO 3 RND 1 - + DUP 0 = IF DROP 1
9|ELSE DUP UGL 0 = IF 1 - THEN THEN DUP 0 1 ROT 4 2 BOX
10|4 +LOOP ;
11|: DRAWGROUND 4 56 0 5 DRGS 192 56 14 DRGS DROP ;
12|-->
13|
14|
15|

```

```

+-----Block      153-----
0|( SPACE INVADERS RACK COORDINATE BUMPER ROUTINE )
1|SUBR INVBUMPER
2|MASTERY LHLD, DMASTERY LDED, 7 D BIT, 0=, IF, INVUL LBCD,
3|ELSE, INVLL LBCD, THEN, FLIPCHECK CALL,
4|0=, IF, DMASTERY SDED, DMASTERX LDED, MASTERX LHLD,
5|D DAD, MASTERX SHLD, H A MOV, 5 CPI, CY, IF,
6|1 A MVI, GAMEOVER STA, THEN, 0 INVADERDROPAT H LXI,
7|ELSE, D DAD, MASTERY SHLD, 0 INVADERPAT H LXI,
8|THEN, INVPATAB SHLD, RET,
9|-->
10|
11|
12|
13|
14|
15|
+-----Block      154-----
0|( LOCAL FORCE FIELD GOODIES )
1|SUBR eraseff FFLAG LDA, A ANA, RZ, A XRA, FFLAG STA,
2|20 A MVI, FFTIMER STA, FIELDRAW CALL, RET,
3|DECIMAL -->
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      155-----
0|( FIREBASE STUFF FOR SPACE INVADERS )
1|HEX SUBR SIFBINTER EXPLODEFB CALL, RET,
2|HEX DATA SIFBA ASM SIFBINTER SETI 0C05 B005 SETDDC
3|PLAYERANIM AJMP
4|DECIMAL -->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      156-----
0|( GOODIES TO EXPLODE A BOMB )
1|DATA BOMBEXP 2 B, 5 B, QUAD 1010 B, 0000 B, 0100 B, 0000 B,
2|1111 B, 0000 B, 0100 B, 0000 B, 1001 B, 0000 B,
3|DECIMAL
4|DATA ABEXP ASM 0 0 SETDC BOMBEXP SETP 6 SWAIT NULPAT
5|SETP AHALT
6|( ROUTINE TO EXPLODE A BOMB )
7|SUBR BANGBANG ABEXP H LXI, CRASHA CALL, XAWRITE H LXI,
8|L PQRL X STX, H PQRH X STX, RET,
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      157-----
0|( CHECK FOR BOMB INTERCEPT ) HEX
1|F= CKPHASOR F= CKFF F= FFSL F= FFOK F= FFZL
2|SUBR INTERBOMB <ASSEMBLE
3|0 FBVECTOR Y LXIX, CHECKVEC CALL, CKPHASOR JRZ,
4|Y PUSHX, XTIX, EXPLODEFB CALL, X POPX, BANGBANG JMP,
5|LABEL CKPHASOR PVI LIYD, CHECKVEC CALL, CKFF JRZ,
6|PQSRH PQS Y RESX, BANGBANG JMP,
7|LABEL CKFF FFLAG LDA, A ANA, BANGBANG JZ, VYH X C LDX, 0 B MVI,
8|0 FIELDADR H LXI, B DAD, L E MOV, H D MOV, 3 B MVI,
9|LABEL FFSL M A MOV, A ANA, FFOK JRNZ, H INX, FFSL DJNZ,
10|BANGBANG JMP,
11|LABEL FFOK RRC, RRC, 3F ANI, VXH X SUBX,
12|4 ADI, 7 CPI, BANGBANG JNC, C DCR, D DCX, D PUSH,
13|C L MOV, 0 H MVI, H DAD, H DAD, H DAD, H DAD,
14|L C MOV, H B MOV, H DAD, H DAD, B DAD, -->
15|
+-----Block      158-----
0|( BOMB INTERCEPT CHECKER CONTINUED )
1|5 B MVI,
2|LABEL FFZL XTHL, M A MOV, A ANA, 0<>, IF,
3|A C MOV, 3 ANI, 20 ORI, MAGIC OUT, A XRA, A M MOV,
4|H INX, XTHL, C A MOV, RRC, RRC,
5|3F ANI, A E MOV, 0 D MVI, XCHG, D DAD, 0FF M MVI,
6|H INX, 0 M MVI, XCHG, ELSE, H INX, XTHL, THEN,
7|50 D LXI, D DAD, FFZL DJNZ,
8|H POP, BANGBANG JMP,
9|ASSEMBLE>
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      159-----
0|(< PHASOR INTERCEPT CHECK ROUTINE )
1|DECIMAL F= INTLOG
2|SUBR PINTER <ASSEMBLE
3|PINTERFLAG LDA, A ANA, RNZ,
4|3 C MVI, CHECKALL CALL, 0<>, IF,
5|VIDENT Y A LDX, 2 ANI, 0<>, IF, Y PUSHX, XTIX, BANGBANG CALL,
6|X POPX, A XRA, INTLOG JMPR, THEN,
7|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
8|VYL Y L LDX, VYH Y H LDX, PINTERX SHLD,
9|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
10|VRACK Y C LDX, 6 C BIT, 0=, IF, XALIVEBITS CALL, M XRA,
11|A M MOV, THEN, 1 A MVI, INTLOG JMPR,
12|THEN, RACKCHECK CALL, RZ, 2 A MVI,
13|LABEL INTLOG PINTERFLAG STA, C A MOV, PINTERN STA,
14|verase CALL, PQSRH PQS X RESX,
15|RET, ASSEMBLE> -->

```

```

+-----Block      160-----
0|(< BOMB ANIMATION SCORES )
1|HEX
2|DATA AB1TBL BOMB1 , BOMB2 , BOMB3 , BOMB4 ,
3|DATA AB2TBL TOMB1 , TOMB2 , TOMB3 , TOMB4 ,
4|DATA ABOMBSUB ASM XIWRITE SETR NULPAT SETFP INTERBOMB SETI
5|-80 0 SETDC
6|FOREVER 0 PATI 6 SWAIT 2 PATI 6 SWAIT 4 PATI 6 SWAIT
7|6 PATI 6 SWAIT 4 PATI 6 SWAIT 2 PATI 6 SWAIT EVERFOR
8|DATA ABD1 ASM AB1TBL SETPT ABOMBSUB AJMP
9|DATA ABD2 ASM AB2TBL SETPT ABOMBSUB AJMP
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      161-----
0|(< SUBROUTINE TO DROP A BOMB IF POSSIBLE )
1|HEX F= BITSCL F= BITFND
2|SUBR BOMBCHECK <ASSEMBLE BOMBTIMER LDA, A ANA, RNZ,
3|LDAR, 0F ANI, A C MOV, 8 ANI, 0<>, IF, MASTERY 1 + LDA,
4|A B MOV, VYH FBVECTOR LDA, B SUB, CY~, IF, RRC, RRC,
5|RRC, RRC, 7 ANI, A C MOV, ELSE, 3 C RES, THEN, THEN,
6|0 B MVI, 0 RACKBITS H LXI, B DAD, M A MOV, A ANA, RZ,
7|LABEL BITSCL RRC, BITFND JRC, B INR, BITSCL JMPR,
8|LABEL BITFND C A MOV, RLC, RLC, RLC, B ORA, A C MOV,
9|CALCIN VX CALL, BOMBDX D LXI, D DAD, MASTERX LDED, D DAD,
10|H PUSH, L RALR, H A MOV, RAL, A B MOV,
11|C A MOV, CALCIN VY CALL, BOMBDY D LXI, D DAD,
12|MASTERY LDED, D DAD, H PUSH,
13|-->
14|
15|

```



```

+-----Block      162-----
0|( BOMBER CONTINUED )
1|LDAR, 1 ANI, 0=, IF, ABD1 H LXI, ELSE, ABD2 H LXI, THEN,
2|H PUSH, ( ALIST ) B A MOV, 8 SUI, A L MOV, 0 H MVI, H PUSH,
3|2A4 H LXI, H PUSH,
4|INVADERSLEFT LDA, A C MOV, SKILLFACTOR LDA, A ANA, 0=, IF,
5|LDAR, 1F ANI, ELSE, A XRA, THEN,
6|6 ADI, C ADD, BOMBTIMER STA,
7|XYVSTART JMP, ASSEMBLE>
8|CODE BOMBER X PUSHX, Y PUSHX, EXX, BOMBCHECK CALL,
9|EXX, Y POPX, X POPX, ( SHOVEL INVADERS LEFT )
10|
11|INVADERSLEFT LDA, 5 ADI, TIMEBASE MB2 + STA,
12|
13|NEXT
14|DECIMAL -->
15|

```

```

+-----Block      163-----
0|( GORF BOUNCE ANIMATION )
1|HEX
2|DATA BOUNCER ASM 0 100 SETDC -10 0 SETDDC GORF SETP 11 SWAIT
3|GORFB SETP 120 100 SETDC 11 SWAIT ARET
4|DATA BOUNCEL ASM 0 -100 SETDC -10 0 SETDDC GORF SETP 11 SWAIT
5|GORFB SETP 120 -100 SETDC 11 SWAIT ARET
6|DATA BR3 ASM 3 AREPEAT BOUNCER ACALL ALOOP ARET
7|DATA BL3 ASM 3 AREPEAT BOUNCEL ACALL ALOOP ARET
8|DATA BL5 ASM 5 AREPEAT BOUNCEL ACALL ALOOP ARET
9|DATA SETLVL ASM NULPAT SETFP XADDWRITE SETR 4200 SETXC ARET
10|DATA BOUNCE ASM SETLVL ACALL B000 SETYC BL5 ACALL
11|BR3 ACALL BL3 ACALL BR3 ACALL BL3 ACALL NULPAT SETP AHALT
12|DATA GORFL ASM SETLVL ACALL B000 SETYC BL5 ACALL AHALT
13|DATA GORFR ASM SETLVL ACALL 0 SETYC 5 AREPEAT BOUNCER ACALL
14|ALoop AHALT
15|DECIMAL -->

```

```

+-----Block      164-----
0|( UFO1 ANIMATION )
1|HEX
2|DATA UFO1L ASM XADDWRITE SETR 4280 SETXC 0 SETYC
3|NULPAT SETFP
4|UFO2 SETP 0 200 SETDC 20 SWAIT 0 -8 SETDDC 40 SWAIT 0 0 SETDC
5|0 0 SETDDC 18 SWAIT 0 -8 SETDDC 40 SWAIT 0 0 SETDDC 3F SWAIT
6|AHALT
7|DATA UFO1R ASM XADDWRITE SETR 4280 SETXC B400 SETYC
8|NULPAT SETFP UFO2 SETP 0 -200 SETDC 20 SWAIT 0 8 SETDDC
9|40 SWAIT 0 0 SETDC 0 0 SETDDC 18 SWAIT 0 8 SETDDC 40 SWAIT
10|0 0 SETDDC 3F SWAIT AHALT
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      165-----
0|( ANIMATION SEQUENCES FOR UFO )
1|HEX
2|DATA UFOL ASM UFO SETP NULPAT SETFP 4100 SETXC 0 SETYC
3|0 100 SETDC 78 SWAIT 3C SWAIT AHALT
4|DATA UFOR ASM UFO SETP NULPAT SETFP 4100 SETXC 0B400 SETYC
5|0 -100 SETDC 78 SWAIT 3C SWAIT AHALT
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      166-----
0|( UFO ANIMATION CONTINUED )
1|HEX DATA UFOATBL
2|43 B, UFOL , 43 B, UFOL , 43 B, UFOR , 43 B, UFOR ,
3|45 B, UF01L , 45 B, UF01L , 45 B, UF01R , 43 B, UFOR ,
4|44 B, GORFL , 44 B, GORFR , 45 B, UF01R , 43 B, UFOL ,
5|44 B, GORFL , 44 B, GORFR , 43 B, UFOL , 43 B, UFOR ,
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      167-----
0|( SUBROUTINE TO SEND OVER UFOS )
1|HEX SUBR UFOCHECKSUBR
2|UFOTIMER LDA, A ANA, RNZ,
3|LDAR, 7F ANI, 80 ADI, UFOTIMER STA, LDAR, 0F ANI,
4|A C MOV, RLC, C ADD, A C MOV, 0 B MVI, UFOATBL H LXI,
5|B DAD, M C MOV, H INX, M E MOV, H INX, M D MOV,
6|0 H LXI, H PUSH, H PUSH, D PUSH, B PUSH, 1A6 H LXI,
7|H PUSH, XYVSTART JMP,
8|CODE UFOCHECK X PUSHX, Y PUSHX, EXX, UFOCHECKSUBR CALL,
9|EXX, Y POPX, X POPX, NEXT
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      168-----
0|( SPACE INVADERS INITIALIZATION ) HEX
1|DATA INVCOLORS 12 B, 7D B, E4 B, A3 B, 12 B, 7D B, 5A B, 0F B,
2|: SISPELL 100 5000 428 A" SPACE INVADERS" COUNT SPOST ;
3|: INITSPACEINV 0 FLOOD INITMISSIONRAM CL MAKEPATS 31 MISSION !
4|DRAWMISSIONSCREEN SISPELL DRAWGROUND 3 9 OUTP
5|RESETRACK 20 INVADERSLEFT ! SKILLFACTOR @ IF 2400
6|ELSE 2800 THEN MASTERX ! FD00 DMASTERX !
7|8 0 DO 0 I RACKBITS B! 0F I ALIVEBITS B! LOOP
8|80 0 DO MASTERY @ I ANIMSTATE ! MASTERX @ I 1+
9|ANIMSTATE ! 2 +LOOP
10|0 FFLAG ! -2 INITFF
11|INVBUMPER BUMPMASSTERROUTINE ! 0 INVADERPAT INVPATAB !
12|' TH REINIT ! PINTER PHASINTR ! eraseff FIREACTION !
13|SIFBA FBANIM ! PINTERFLAG ZERO
14|INVNORMLPAT NORMLP1 ! GETNODE DUP PV1 ! 0 SWAP ! ;
15|DECIMAL -->

+-----Block      169-----
0|( CRAB RACK ENTRY ANIMATION )
1|DATA RENT ASM 11 SWAIT 0 0 SETDC RENTGAL SETR 120 SWAIT AHALT
2|DATA ACRAB1 ASM CRAB1A SETP RENT AJMP
3|DATA ACRAB2 ASM CRAB2A SETP RENT AJMP
4|DATA ACRAB3 ASM CRAB3A SETP RENT AJMP
5|TABLE SIDOTBL ACRAB3 , ACRAB3 , ACRAB2 , ACRAB1 ,
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      170-----
0|( GORF SPIRAL OUT )
1|HEX
2|DATA GORFSPI ASM FEBA 067D SETDC 0007 0603 SETDDC
3|DECIMAL
4|NULPAT SETP SPWRITE SETR NULRET SETI 1 SWAIT
5|GORF1 SETP 40 SWAIT
6|GORF2 SETP 35 SWAIT
7|GORF3 SETP 30 SWAIT
8|GORF4 SETP 25 SWAIT
9|GORF5 SETP 20 SWAIT
10|GORF SETP
11|FOREVER 120 SWAIT EVERFOR
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      171-----
0|( GAME START SPIRAL OUT THE GORF )
1|HEX
2|: UDL 0 DO UPDATEALL BMS LOOP ;
3|: SPOUT 3 INVCOLORS FUC
4|GENLINE LSTART
5|MUTHAX @ MUTHAY @ 2DUP SETLXY
6|SVCY ! SVCX !
7|8 8 50 50 SETSF
8|4 UDL NULRET LINIT !
9|2 SPIRALRATE !
10|GORFSPI 67 0B2 VSTART
11|0E0 UDL ;
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      172-----
0|( SPACE INVADERS DUMP OUT SEQUENCE )
1|HEX 0 V= GORFV
2|: SIWAIT WTIMER ! BEGIN FIRECHECK PHASORINTERCEPTCHECK
3|BMS WTIMER @ 0 = END ;
4|: SIDO GETNODE GORFV ! IASCORE B2MUSIC
5|SPOUT ( SPIRAL OUT GORF ) ACTFB ( START FIRE BASE )
6|0D 9 OUTP
7|BOUNCE 47 1B2 GORFV @ XVSTART 0AA SIWAIT
8|4 0 DO 8 0 DO
9|( 0D AND 13 ARE VXL AND VYL RESPECTIVELY )
10|I 8 * J + 0D GORFV @ + @ 13 GORFV @ + @ 400 +
11|J 400 * MASTERX @ + I 1000 *
12|J SIDOTBL @ 0C 1A2 VMOVE 0A SIWAIT
13|LOOP LOOP 60 SIWAIT DRAWFF ;
14|DECIMAL -->
15|

```

```

+-----Block      173-----
0|( TRANSITION FROM MISSION 1 TO MISSION 2 ) HEX
1|DATA GLTCOLORS 12 B, 7D B, 0B B, 5A B, 12 B, 7D B, 0B B, 5A B,
2|DECIMAL
3|: TRANSIT2 DI
4|SISPELL ERASEFF GLTCOLORS COLOR DI 0 0 4 56 0 BOX
5|0 56 14 136 0 BOX EI
6|44 HORCB OUTP 7 0 OUTP 18 4 OUTP
7|0 44 DO 1 HORCB WVI OUTP EI 3 PWAIT -1 +LOOP
8|WVI 7 4 OUTP EI ;
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```

```

+-----Block 174-----
0|( CRUDE SPACE INVADERS SCAN LOOP )
1|: SISCAN BMS WRTINV FIRECHECK PHASORINTERCEPTCHECK BOMBER
2|DRAWFF UFOCHECK PLAYERHITCHECK ;
3|: SI INITSPACEINV SIDO TH
4|BEGIN SISCAN ENDOFFRAME @ END
5|GAMEOVER @ 0= IF TRANSIT2 THEN ;
6|HEX A5 GSAB U! ' SI GSAB 1+ U!
7|: BEGINGAME STARTGAME SKILLFACTOR ! GSAB 1+ @ DOIT ;
8|DECIMAL ;S
9|
10|
11|
12|
13|
14|
15|
+-----Block 180-----
0|( GORF SPIRAL OUT )
1|HEX
2|DATA GORFSP1 ASM FEBA 067D SETDC 0007 0603 SETDDC
3|DECIMAL
4|NULPAT SETP SPWRITE SETR NULRET SETI 1 SWAIT
5|GORF1 SETP 90 SWAIT
6|GORF2 SETP 80 SWAIT
7|GORF3 SETP 70 SWAIT
8|GORF4 SETP 60 SWAIT
9|GORF5 SETP 50 SWAIT
10|GORF SETP
11|FOREVER 120 SWAIT EVERFOR
12|DECIMAL -->
13|
14|
15|
+-----Block 181-----
0|( GAME START SPIRAL OUT THE GORF )
1|HEX
2|: SPOUT
3|STARTGAME INITSPACEINV
4|GENLINE LSTART
5|2800 6400 SETLXY
6|8 8 80 80 SETSF
7|4 0 DO UPDATEALL LOOP NULRET LINIT !
8|GORFSP1 67 032 VSTART
9|240 0 DO UPDATEALL LOOP ;
10|DECIMAL ;S
11|
12|
13|
14|
15|

```

```

100|( START OF GALAXIANS AREA )
101|( GALAXIAN 1A )
102|( GALAXIAN 1B )
103|( GALAXIAN 2A )
104|( GAL2B )
105|( GALAXIAN 3A )
106|( GALAXIAN 3B )
107|( GALAXIAN 4 )
108|( FIRST ROTATED GALAX3 PATTERN )
109|( SECOND ROTATED GALAX3 PATTERN )
110|( THIRD ROTATED GALAX3 PATTERN )
111|( LAST ROTATED GALAX3 PATTERN )
112|( FIRST ROTATED GALAX2 PATTERN )
113|( SECOND ROTATED GALAX2 PATTERN )
114|( THIRD ROTATED GALAX2 PATTERN )
115|( LAST ROTATED GALAX2 PATTERN )
116|( FIRST ROTATED GALAX1 PATTERN )
117|( SECOND ROTATED GALAX1 PATTERN )
118|( THIRD ROTATED GALAX1 PATTERN )
119|( LAST ROTATED GALAX1 PATTERN )
120|( FIRST ROTATED GALAXIAN 4 )
121|( SECOND ROTATED GALAXIAN 4 )
122|( THIRD ROTATED GALAXIAN 4 )
123|( LAST GALAXIAN 4 ROTATED )
150|( GALAXIANS GAME )
151|( MORE GOODIES ) DECIMAL
152|( BUMP GALAXIAN RACK COORDINATES ) HEX
153|( INTERRUPT BOMB DROPPER ) HEX
154|( INTERRUPT BOMB DROPPER CONTINUED )
155|( START A BOMB DROPPING ) HEX
156|( ANIMATION LISTS TO ACTIVATE FIREBASE AND BOMBING )
157|( SPACE MISSIONS GALAXIAN ATTACK SOUND- GA ) HEX
158|( SPACE MISSIONS BMUSIC BLOCK cont. )
159|( SUBROUTINE TO START AN ATTACKER VECTOR ) DECIMAL
160|( ROUTINE TO RETARGET AN ATTACKER )
161|( PATTERN TABLE FOR GAL3 )
162|( REENTER GALAXIAN 4 )
163|( LEFT ROLL GAL3 )
164|( LEFT ROLL GAL2 )
165|( ROLL GAL1 LEFT AND RIGHT )
166|( RANDOM GORF GOODIES )
167|( LEFT PEELOFF FOR GALAXIAN 4 )
168|( ATTACK PATH TABLES )
169|( SUBROUTINE TO RESET THE ATTACK TIMER )
170|( ATTACK ROUTINE FOR CODES 1 THRU 6 ) HEX
171|( ATTACK ROUTINE FOR CODES 7-10 )
172|( CHECK FOR ATTACK ROUTINE ) HEX
173|( PHASOR INTERCEPT CHECK ROUTINE )
174|( GALAXIAN COLORS AND WAIT ROUTINE )
175|( INITIALIZE GALAXIAN GAME )
176|( SCAN LOOP AND WAIT ROUTINE )
177|( ANIMATION STUFF TO DUMP OUT GALAXIANS )
178|( DUMPOUT ROUTINE )
179|( SCAN LOOP AND STARTUP )
196|( SYSTEM LOAD ROUTINE ) 16 BASE 1
198|( SYSTEM LOAD ROUTINE ) 16 BASE 1
199|( SYSTEM LOAD ROUTINE ) 16 BASE 1

```

```

+-----Block      100-----
0|( START OF GALAXIANS AREA )
1|DATA GSAB 0 B, 0 ,
2|DECIMAL -->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      101-----
0|( GALAXIAN 1A )
1|DECIMAL DATA GAL1A 3 B, 11 B, QUAD
2|3300 B, 1100 B, 0000 B,
3|3330 B, 1000 B, 0000 B,
4|0030 B, 1000 B, 0000 B,
5|0031 B, 1100 B, 0000 B,
6|0111 B, 1311 B, 0000 B,
7|1111 B, 1111 B, 0000 B,
8|0111 B, 1311 B, 0000 B,
9|0031 B, 1100 B, 0000 B,
10|0030 B, 1000 B, 0000 B,
11|3330 B, 1000 B, 0000 B,
12|3300 B, 1100 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      102-----
0|( GALAXIAN 1B )
1|DECIMAL DATA GAL1B 3 B, 11 B, QUAD
2|0033 B, 0111 B, 0000 B,
3|0030 B, 1100 B, 0000 B,
4|0030 B, 1000 B, 0000 B,
5|0031 B, 1100 B, 0000 B,
6|0111 B, 3110 B, 0000 B,
7|1111 B, 1100 B, 0000 B,
8|0111 B, 3110 B, 0000 B,
9|0031 B, 1100 B, 0000 B,
10|0030 B, 1000 B, 0000 B,
11|0030 B, 1100 B, 0000 B,
12|0033 B, 0111 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      103-----
0|( GALAXIAN 2A )
1|DATA GAL2A 3 B, 11 B, QUAD
2|1100 B, 2200 B, 0000 B,
3|1110 B, 2000 B, 0000 B,
4|0110 B, 2000 B, 0000 B,
5|0012 B, 2200 B, 0000 B,
6|0222 B, 1222 B, 0000 B,
7|2222 B, 2200 B, 0000 B,
8|0222 B, 1222 B, 0000 B,
9|0012 B, 2200 B, 0000 B,
10|0110 B, 2000 B, 0000 B,
11|1110 B, 2000 B, 0000 B,
12|1100 B, 2200 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      104-----
0|( GAL2B )
1|DECIMAL DATA GAL2B 3 B, 11 B, QUAD
2|0011 B, 0222 B, 0000 B,
3|0010 B, 2200 B, 0000 B,
4|0010 B, 2000 B, 0000 B,
5|0012 B, 2200 B, 0000 B,
6|0222 B, 1220 B, 0000 B,
7|2222 B, 2200 B, 0000 B,
8|0222 B, 1220 B, 0000 B,
9|0012 B, 2200 B, 0000 B,
10|0010 B, 2000 B, 0000 B,
11|0010 B, 2200 B, 0000 B,
12|0011 B, 0222 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      105-----
0|( GALAXIAN 3A )
1|DATA GAL3A 3 B, 11 B, QUAD
2|2200 B, 3300 B, 0000 B,
3|2220 B, 3000 B, 0000 B,
4|0220 B, 3000 B, 0000 B,
5|0023 B, 3300 B, 0000 B,
6|0333 B, 2333 B, 0000 B,
7|3333 B, 3300 B, 0000 B,
8|0333 B, 2333 B, 0000 B,
9|0023 B, 3300 B, 0000 B,
10|0220 B, 3000 B, 0000 B,
11|2220 B, 3000 B, 0000 B,
12|2200 B, 3300 B, 0000 B,
13|DECIMAL -->
14|
15|

```



```

+-----Block      106-----
0|( GALAXIAN 3B )
1|DECIMAL DATA GAL3B 3 B, 11 B, QUAD
2|0022 B, 0333 B, 0000 B,
3|0020 B, 3300 B, 0000 B,
4|0020 B, 3000 B, 0000 B,
5|0023 B, 3300 B, 0000 B,
6|0333 B, 2330 B, 0000 B,
7|3333 B, 3300 B, 0000 B,
8|0333 B, 2330 B, 0000 B,
9|0023 B, 3300 B, 0000 B,
10|0020 B, 3000 B, 0000 B,
11|0020 B, 3300 B, 0000 B,
12|0022 B, 0333 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      107-----
0|( GALAXIAN 4 )
1|DATA GAL4 4 B, 11 B, QUAD
2|0000 B, 0222 B, 2200 B, 0000 B,
3|0000 B, 2211 B, 0000 B, 0000 B,
4|0002 B, 2113 B, 0000 B, 0000 B,
5|0022 B, 1113 B, 3000 B, 0000 B,
6|0000 B, 0111 B, 3300 B, 0000 B,
7|1111 B, 1133 B, 3330 B, 0000 B,
8|0000 B, 0111 B, 3300 B, 0000 B,
9|0022 B, 1113 B, 3000 B, 0000 B,
10|0002 B, 2113 B, 0000 B, 0000 B,
11|0000 B, 2211 B, 0000 B, 0000 B,
12|0000 B, 0222 B, 2200 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      108-----
0|( FIRST ROTATED GALAX3 PATTERN )
1|DECIMAL DATA GAL3R1 4 B, 12 B, QUAD
2|0003 B, 3000 B, 0000 B, 0 B,
3|0003 B, 0000 B, 0000 B, 0 B,
4|0003 B, 0030 B, 0000 B, 0 B,
5|2203 B, 3300 B, 0000 B, 0 B,
6|2223 B, 2330 B, 3000 B, 0 B,
7|2023 B, 3333 B, 0000 B, 0 B,
8|0003 B, 3323 B, 0000 B, 0 B,
9|0003 B, 3333 B, 0000 B, 0 B,
10|0000 B, 0233 B, 3030 B, 0 B,
11|0000 B, 0223 B, 0330 B, 0 B,
12|0000 B, 0220 B, 0330 B, 0 B,
13|0000 B, 0200 B, 0000 B, 0 B,
14|-->
15|

```

```

+-----Block      109-----
0|( SECOND ROTATED GALAX3 PATTERN )
1|DECIMAL DATA GAL3R2 4 B, 12 B, QUAD
2|0003 B, 0000 B, 0000 B, 0 B,
3|0030 B, 0000 B, 0000 B, 0 B,
4|0003 B, 0003 B, 0000 B, 0 B,
5|0000 B, 3330 B, 0000 B, 0 B,
6|0220 B, 3233 B, 0300 B, 0 B,
7|2222 B, 3333 B, 3000 B, 0 B,
8|0003 B, 3332 B, 3000 B, 0 B,
9|0003 B, 3333 B, 3000 B, 0 B,
10|0003 B, 3320 B, 0303 B, 0 B,
11|0000 B, 0022 B, 0030 B, 0 B,
12|0000 B, 0022 B, 0000 B, 0 B,
13|0000 B, 0020 B, 0000 B, 0 B,
14|DECIMAL -->
15|

```

```

+-----Block      110-----
0|( THIRD ROTATED GALAX3 PATTERN )
1|DECIMAL DATA GAL3R3 4 B, 11 B, QUAD
2|0330 B, 0000 B, 0000 B, 0 B,
3|0300 B, 0000 B, 0000 B, 0 B,
4|0030 B, 0003 B, 0000 B, 0 B,
5|0033 B, 3330 B, 0000 B, 0 B,
6|0223 B, 3233 B, 0300 B, 0 B,
7|2222 B, 3333 B, 3000 B, 0 B,
8|0000 B, 3332 B, 3003 B, 0 B,
9|0000 B, 3333 B, 3333 B, 0 B,
10|0000 B, 0022 B, 0000 B, 0 B,
11|0000 B, 0002 B, 2000 B, 0 B,
12|0000 B, 0022 B, 2000 B, 0 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      111-----
0|( LAST ROTATED GALAX3 PATTERN )
1|DECIMAL DATA GAL3R4 4 B, 8 B, QUAD
2|0000 B, 0303 B, 0000 B, 0 B,
3|0000 B, 0303 B, 0000 B, 0 B,
4|0300 B, 3333 B, 3003 B, 0 B,
5|0333 B, 3232 B, 3333 B, 0 B,
6|0000 B, 3333 B, 3000 B, 0 B,
7|0022 B, 2333 B, 2220 B, 0 B,
8|0222 B, 0333 B, 0222 B, 0 B,
9|0220 B, 0030 B, 0222 B, 0 B,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      112-----
0|( FIRST ROTATED GALAX2 PATTERN )
1|DECIMAL DATA GAL2R1 4 B, 12 B, QUAD
2|0002 B, 2000 B, 0000 B, 0 B,
3|0002 B, 0000 B, 0000 B, 0 B,
4|0002 B, 0020 B, 0000 B, 0 B,
5|1102 B, 2200 B, 0000 B, 0 B,
6|1112 B, 1220 B, 2000 B, 0 B,
7|1012 B, 2222 B, 0000 B, 0 B,
8|0002 B, 2212 B, 0000 B, 0 B,
9|0002 B, 2222 B, 0000 B, 0 B,
10|0000 B, 0222 B, 2020 B, 0 B,
11|0000 B, 0222 B, 0220 B, 0 B,
12|0000 B, 0220 B, 0220 B, 0 B,
13|0000 B, 0200 B, 0000 B, 0 B,
14|DECIMAL -->
15|

```

```

+-----Block      113-----
0|( SECOND ROTATED GALAX2 PATTERN )
1|DECIMAL DATA GAL2R2 4 B, 12 B, QUAD
2|0002 B, 0000 B, 0000 B, 0 B,
3|0020 B, 0000 B, 0000 B, 0 B,
4|0002 B, 0002 B, 0000 B, 0 B,
5|0000 B, 2220 B, 0000 B, 0 B,
6|0110 B, 2122 B, 0200 B, 0 B,
7|1111 B, 2222 B, 2000 B, 0 B,
8|0002 B, 2221 B, 2000 B, 0 B,
9|0002 B, 2222 B, 2000 B, 0 B,
10|0002 B, 2210 B, 0202 B, 0 B,
11|0000 B, 0011 B, 0020 B, 0 B,
12|0000 B, 0011 B, 0000 B, 0 B,
13|0000 B, 0010 B, 0000 B, 0 B,
14|DECIMAL -->
15|

```

```

+-----Block      114-----
0|( THIRD ROTATED GALAX2 PATTERN )
1|DECIMAL DATA GAL2R3 4 B, 11 B, QUAD
2|0220 B, 0000 B, 0000 B, 0 B,
3|0200 B, 0000 B, 0000 B, 0 B,
4|0020 B, 0002 B, 0000 B, 0 B,
5|0022 B, 2220 B, 0000 B, 0 B,
6|0112 B, 2122 B, 0200 B, 0 B,
7|1111 B, 2222 B, 2000 B, 0 B,
8|0000 B, 2221 B, 2002 B, 0 B,
9|0000 B, 2222 B, 2222 B, 0 B,
10|0000 B, 0011 B, 0000 B, 0 B,
11|0000 B, 0001 B, 1000 B, 0 B,
12|0000 B, 0011 B, 1000 B, 0 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      115-----
0|( LAST ROTATED GALAX2 PATTERN )
1|DECIMAL DATA GAL2R4 4 B, 8 B, QUAD
2|0000 B, 0202 B, 0000 B, 0 B,
3|0000 B, 0202 B, 0000 B, 0 B,
4|0200 B, 2222 B, 2002 B, 0 B,
5|0222 B, 2121 B, 2222 B, 0 B,
6|0000 B, 2222 B, 2000 B, 0 B,
7|0011 B, 1222 B, 1110 B, 0 B,
8|0111 B, 0222 B, 0111 B, 0 B,
9|0110 B, 0020 B, 0011 B, 0 B,
10|DECIMAL -->
11|
12|
13|
14|
15|
+-----Block      116-----
0|( FIRST ROTATED GALAX1 PATTERN )
1|DECIMAL DATA GAL1R1 4 B, 12 B, QUAD
2|0001 B, 1000 B, 0000 B, 0 B,
3|0001 B, 0000 B, 0000 B, 0 B,
4|0001 B, 0010 B, 0000 B, 0 B,
5|3301 B, 1100 B, 0000 B, 0 B,
6|3331 B, 3110 B, 1000 B, 0 B,
7|3031 B, 1111 B, 0000 B, 0 B,
8|0001 B, 1131 B, 0000 B, 0 B,
9|0001 B, 1111 B, 0000 B, 0 B,
10|0000 B, 0111 B, 1010 B, 0 B,
11|0000 B, 0111 B, 0110 B, 0 B,
12|0000 B, 0110 B, 0110 B, 0 B,
13|0000 B, 0100 B, 0000 B, 0 B,
14|DECIMAL -->
15|
+-----Block      117-----
0|( SECOND ROTATED GALAX1 PATTERN )
1|DECIMAL DATA GAL1R2 4 B, 12 B, QUAD
2|0001 B, 0000 B, 0000 B, 0 B,
3|0010 B, 0000 B, 0000 B, 0 B,
4|0001 B, 0001 B, 0000 B, 0 B,
5|0000 B, 1110 B, 0000 B, 0 B,
6|0330 B, 1311 B, 0100 B, 0 B,
7|3333 B, 1111 B, 1000 B, 0 B,
8|0001 B, 1110 B, 1000 B, 0 B,
9|0001 B, 1111 B, 1000 B, 0 B,
10|0001 B, 1130 B, 0100 B, 0 B,
11|0000 B, 0000 B, 0010 B, 0 B,
12|0000 B, 0000 B, 0000 B, 0 B,
13|0000 B, 0000 B, 0000 B, 0 B,
14|DECIMAL -->
15|

```

```

+-----Block      118-----
0|( THIRD ROTATED GALAX1 PATTERN )
1|DECIMAL DATA GAL1R3 4 B, 11 B, QUAD
2|0110 B, 0000 B, 0000 B, 0 B,
3|0100 B, 0000 B, 0000 B, 0 B,
4|0010 B, 0001 B, 0000 B, 0 B,
5|0011 B, 1110 B, 0000 B, 0 B,
6|0331 B, 1311 B, 0100 B, 0 B,
7|3333 B, 1111 B, 1000 B, 0 B,
8|0000 B, 1113 B, 1001 B, 0 B,
9|0000 B, 1111 B, 1111 B, 0 B,
10|0000 B, 0033 B, 0000 B, 0 B,
11|0000 B, 0003 B, 3000 B, 0 B,
12|0000 B, 0033 B, 3000 B, 0 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      119-----
0|( LAST ROTATED GALAX1 PATTERN )
1|DECIMAL DATA GAL1R4 4 B, 8 B, QUAD
2|0000 B, 0101 B, 0000 B, 0 B,
3|0000 B, 0101 B, 0000 B, 0 B,
4|0100 B, 1111 B, 1001 B, 0 B,
5|0111 B, 1313 B, 1111 B, 0 B,
6|0000 B, 1111 B, 1000 B, 0 B,
7|0033 B, 3111 B, 3330 B, 0 B,
8|0333 B, 0111 B, 0333 B, 0 B,
9|0330 B, 0010 B, 0033 B, 0 B,
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      120-----
0|( FIRST ROTATED GALAXIAN 4 )
1|DATA GAL4R1 4 B, 11 B, QUAD
2|0000 B, 2220 B, 0000 B, 0000 B,
3|0022 B, 2000 B, 0000 B, 0000 B,
4|0021 B, 1130 B, 0000 B, 0000 B,
5|0211 B, 1133 B, 3000 B, 0000 B,
6|0211 B, 1113 B, 3300 B, 0000 B,
7|0000 B, 1131 B, 3000 B, 0000 B,
8|0011 B, 1111 B, 3000 B, 0000 B,
9|0110 B, 0111 B, 3000 B, 0000 B,
10|1000 B, 0111 B, 0000 B, 0000 B,
11|0002 B, 2200 B, 2200 B, 0000 B,
12|0000 B, 0222 B, 0000 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      121-----
0|( SECOND ROTATED GALAXIAN 4 )
1|DATA GAL4R2 4 B, 11 B, QUAD
2|0002 B, 0000 B, 0000 B, 0000 B,
3|0020 B, 0000 B, 0000 B, 0000 B,
4|0210 B, 0000 B, 0000 B, 0000 B,
5|2113 B, 3333 B, 0000 B, 0000 B,
6|2111 B, 1133 B, 0000 B, 0000 B,
7|2111 B, 1313 B, 0000 B, 0000 B,
8|2101 B, 0113 B, 0000 B, 0000 B,
9|2001 B, 1113 B, 0020 B, 0000 B,
10|0010 B, 0111 B, 0200 B, 0000 B,
11|0100 B, 1111 B, 2000 B, 0000 B,
12|1002 B, 2222 B, 0000 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      122-----
0|( THIRD ROTATED GALAXIAN 4 )
1|DATA GAL4R3 4 B, 11 B, QUAD
2|0020 B, 0000 B, 0000 B, 0000 B,
3|0200 B, 0030 B, 0000 B, 0000 B,
4|0203 B, 3333 B, 0000 B, 0000 B,
5|2111 B, 1133 B, 0000 B, 0000 B,
6|2111 B, 1313 B, 3020 B, 0000 B,
7|2211 B, 1111 B, 1020 B, 0000 B,
8|0200 B, 1111 B, 1220 B, 0000 B,
9|0200 B, 1011 B, 1200 B, 0000 B,
10|0001 B, 1011 B, 2200 B, 0000 B,
11|0001 B, 0022 B, 0000 B, 0000 B,
12|0010 B, 0000 B, 0000 B, 0000 B,
13|DECIMAL -->
14|
15|

```

```

+-----Block      123-----
0|( LAST GALAXIAN 4 ROTATED )
1|DATA GAL4R4 4 B, 11 B, QUAD
2|0000 B, 0300 B, 0000 B, 0000 B,
3|2000 B, 3330 B, 0020 B, 0000 B,
4|2003 B, 3333 B, 0020 B, 0000 B,
5|2133 B, 1313 B, 3120 B, 0000 B,
6|2111 B, 1311 B, 1120 B, 0000 B,
7|2211 B, 1111 B, 1220 B, 0000 B,
8|0221 B, 0101 B, 2200 B, 0000 B,
9|0022 B, 0102 B, 2000 B, 0000 B,
10|0002 B, 0102 B, 0000 B, 0000 B,
11|0000 B, 0100 B, 0000 B, 0000 B,
12|0000 B, 0100 B, 0000 B, 0000 B,
13|DECIMAL ;S
14|
15|

```

```

+-----Block      150-----
0|( GALAXIANS GAME )
1|-->
2|
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      151-----
0|( MORE GOODIES ) DECIMAL
1|DATA GALAXNORMLPAT GAL1A , GAL1A , GAL2A , GAL3A , GAL4 ,
2|0 , 0 , 0 , GAL1B , GAL1B , GAL2B , GAL3B , GAL4 ,
3|5 ARRAY GALXPAT
4|46 BARRAY GAL1AB 46 BARRAY GAL2AB 46 BARRAY GAL3AB
5|60 BARRAY GAL4AB
6|HEX : MAKEPATS CL 0 0 GAL4 20 WRITEP 0 200 GAL4 20 WRITEP
7|C D 0 GAL4AB 0 0 SNAP 0 GAL4AB 4 GALXPAT !
8|1000 1000 GAL1A 20 WRITEP 1000 1200 GAL1B
9|20 WRITEP 6 D 0 GAL1AB 1000 1000 SNAP 0 GAL1AB DUF 0 GALXPAT !
10|1 GALXPAT !
11|2000 1000 GAL2A 20 WRITEP 2000 1200 GAL2B 20 WRITEP
12|6 D 0 GAL2AB 2000 1000 SNAP 0 GAL2AB 2 GALXPAT !
13|3000 1000 GAL3A 20 WRITEP 3000 1200 GAL3B 20 WRITEP
14|6 D 0 GAL3AB 3000 1000 SNAP 0 GAL3AB 3 GALXPAT ! ;
15|-->

```

```

+-----Block      152-----
0|( BUMP GALAXIAN RACK COORDINATES ) HEX
1|SUBR GALBUMPER MASTERY LHLD, DMASTERY LDED, 7 D BIT, 0=, IF,
2|INVUL LBCD, ELSE, INVLL LBCD, THEN, FLIPCHECK CALL,
3|0=, IF, DMASTERY SDED, ELSE, D DAD, MASTERY SHLD, THEN,
4|RELMT CALL, RET,
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      153-----
0|( INTERRUPT BOMB DROPPER ) HEX
1|F= TBBLP F= DROPLP F= NODROP F= OKDROP F= NOBOMB F= NOBOMB1
2|SUBR BOMBDROPPER <ASSEMBLE
3|20 A MVI, MAGIC OUT, PQTB X A LDX, 0 PQTB X MVIX,
4|LABEL TBBLP PSW PUSH,
5| 0 BOMBARRAY H LXI,
6|NBOMBS A MVI,
7|LABEL DROPLP PSW PUSH, M C MOV, C A MOV,
8|A ANA, NOBOMB JRZ, 055 XRI, A M MOV, 5 D LXI, D DAD, M D MOV,
9|C A MOV,
10|H DCX, M E MOV, D STAX, 05 CPI, 0=, IF, 050 A MVI, D STAX,
11|H INX, H INX, NOBOMB1 JMPR, THEN,
12|H DCX, M B MOV, H DCX, M C MOV, XCHG,
13|B DAD, XCHG, H DCX, M DCR, M A MOV, 3 CPI,
14|NODROP JRC, 6 D BIT, OKDROP JRZ,
15|-->

```

```

+-----Block      154-----
0|( INTERRUPT BOMB DROPPER CONTINUED )
1|LABEL NODROP H DCX, 0 M MVI, NOBOMB JMPR,
2|LABEL OKDROP H INX, H INX, H INX, 05 A MVI, D STAX,
3|E M MOV, H INX, D M MOV, H INX, NOBOMB1 JMPR,
4|LABEL NOBOMB BOMBASIZE D LXI, D DAD,
5|LABEL NOBOMB1 PSW POP, A DCR, DROPLP JRNZ,
6|PSW POP, A DCR, TBBLP JRNZ,
7|RET,
8|ASSEMBLE>
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      155-----
0|( START A BOMB DROPPING ) HEX
1|F= BOMBSL F= BOMBFND
2|SUBR BOMBADIER <ASSEMBLE PQSFRZ PQS X BITX, RNZ,
3|H PUSH, 0 BOMBARRAY H LXI, NBOMBS B MVI, BOMBASIZE D LXI,
4|LABEL BOMBSL M A MOV, A ANA, BOMBFND JRZ, D DAD, BOMBSL DJNZ,
5|H POP, RET,
6|LABEL BOMBFND 05 M MVI, H INX, VXH X A LDX, A M MOV, H INX,
7|VYH X A LDX, A SRLR, A SRLR, A C MOV, VYH FBVECTOR LDA,
8|A SRLR, A SRLR, C SUB, 0<, IF, 0FD CPI, CY~, IF,
9|-1 D LXI, ELSE, -5 D LXI, THEN,
10|ELSE, 3 CPI, CY, IF, -4 D LXI, ELSE, 4F D LXI,
11|THEN, THEN, E M MOV, H INX, D M MOV, H INX, XCHG,
12|VSAL X L LDX, VSAH X H LDX, 1E0 B LXI, 7 VMAGIC X BITX,
13|0=, IF, B DAD, ELSE, A XRA, B DSBC, THEN, 20 A MVI,
14|MAGIC OUT, 05 M MVI, XCHG, E M MOV, H INX, D M MOV,
15|H POP, RET, ASSEMBLE> DECIMAL -->

```



```

+-----Block      156-----
0|( ANIMATION LISTS TO ACTIVATE FIREBASE AND BOMBING )
1|SUBR GALINTER CKATRS CALL, EXPLODEFB CALL, RET,
2|HEX
3|DATA GALFBA ASM GALINTER SETI 1805 B005 SETDDC PLAYERANIM AJMP
4|( BOMB GOODIES )
5|DATA INITBOMBS ASM BOMBDROPPER SETR NULPAT SETP 2 SWAIT
6|DECIMAL
7|DATA BOMBR ASM 10 SWAIT BOMBADIER ASMCALL 20 SWAIT BOMBADIER
8|ASMCALL ARET -->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      157-----
0|( SPACE MISSIONS GALAXIAN ATTACK SOUND- GA ) HEX
1|DATA GASCORE ASM
2| #FS3 #E3 #G2 TONES 1 -2 3F MOVESOUND
3| 10 MASTER 3 -1 20 8 RAMBLE 1 COUNTLIMITS
4| 18 NOISE 0 VIBS AA ABVOLS 2A MCVOLS
5| PLAY 42 VIBS RERAMBLE 1 COUNTLIMITS
6| PLAY 3 1 30 20 RAMBLE 44 VIBS 1 COUNTLIMITS
7| PLAY 3 1 40 1C RAMBLE 4A VIBS 2 COUNTLIMITS
8| PLAY 4 -1 1C 18 RAMBLE PLAY
9|-->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      158-----
0|( SPACE MISSIONS BMUSIC BLOCK cont. )
1|SUBR GA GASCORE H LXI, 0 MUSIC-BARRAY-2 Y LXIX, bmusic JMP,
2|DECIMAL -->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      159-----
0|( SUBROUTINE TO START AN ATTACKER VECTOR ) DECIMAL
1|F= DINGBAT
2|SUBR ATSTART <ASSEMBLE DI, PINTERFLAG LDA, A ANA, DINGBAT JRNZ,
3|H PUSH, B PUSH, 418 D LXI, D PUSH,
4|getnode CALL, H PUSH,
5|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX,
6|CLRVEC CALL, 7 Y A LDX, A VFYBH X STX, 6 Y C LDX,
7|XRACKBITS CALL, M XRA, A M MOV, EI, Y PUSHX, GETASTATE CALL,
8|Y POPX, L VYL X STX, H VYH X STX, E VXL X STX, D VXH X STX,
9|SETSTDW CALL, STARTVEC CALL,
10|UNFRAME B POP, B POP, B POP, H POP,
11|TOGGLEMEMBER CALL, GA JMP,
12|LABEL DINGBAT EI, RET, ASSEMBLE>
13|CODE ATT X PUSHX, H POP, Y PUSHX, D POP, EXX,
14|B POP, H POP, ATSTART CALL,
15|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT -->
+-----Block      160-----
0|( ROUTINE TO RETARGET AN ATTACKER )
1|HEX
2|SUBR TARGET H PUSH, VYH X A LDX, VFYBH X SUBX,
3|A SRLR, A SRLR, A C MOV, VYH FBVECTOR LDA, A SRLR, A SRLR,
4|C SUB, A SRAR, A SRAR, A E MOV, VDYH X B LDX, B SUB, A C MOV,
5|E A MOV, B XRA, C A MOV, 0<, IF, A SRAR, C ADD, THEN,
6|A VDDYL X STX, 7 A BIT, 0 A MVI,
7|0<>, IF, CMA, THEN, A VDDYH X STX,
8|VDDYL X A LDX, AABS CALL, 0E ANI, 6 CPI, CY~, IF, 6 A MVI,
9|THEN, A C MOV, 0 B MVI, VPTBL X L LDX, VPTBH X H LDX,
10|B DAD, M E MOV, H INX, M D MOV, E VPATL X STX,
11|D VPATH X STX, H POP, RET,
12|DECIMAL -->
13|
14|
15|
+-----Block      161-----
0|( PATTERN TABLE FOR GAL3 )
1|DATA GAL3TBL GAL3A , GAL3R1 , GAL3R2 , GAL3R3 , GAL3R4 ,
2|( PATTERN TABLE FOR GAL2 )
3|DATA GAL2TBL GAL2A , GAL2R1 , GAL2R2 , GAL2R3 , GAL2R4 ,
4|( PATTERN TABLE FOR GAL1 )
5|DATA GAL1TBL GAL1A , GAL1R1 , GAL1R2 , GAL1R3 , GAL1R4 ,
6|( PATTERN TABLE FOR GAL4 )
7|DATA GAL4TBL GAL4 , GAL4R1 , GAL4R2 , GAL4R3 , GAL4R4 ,
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      162-----
0|( REENTER GALAXIAN 4 )
1|DECIMAL
2|DATA REENTER4 ASM 19200 SETXC NULPAT SETP 0 0 SETDC 0 0 SETDDC
3|25 SWAIT RENTGAL SETR 2 SWAIT 0 PATI 4 SWAIT FLIPOVER ACALL
4|120 SWAIT AHALT
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      163-----
0|( LEFT ROLL GAL3 )
1|DATA DIVE3 ASM TARGET ASMCALL BOMBR ACALL 30 SWAIT TARGET
2|ASMCALL 40 SWAIT TARGET ASMCALL 40 SWAIT REENTER AJMP
3|DATA LEFT3 ASM GAL3TBL SETPT LEFTROLL ACALL DIVE3 AJMP
4|DATA RIGHT3 ASM GAL3TBL SETPT RIGHTROLL ACALL DIVE3 AJMP
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      164-----
0|( LEFT ROLL GAL2 )
1|DATA DIVE2 ASM TARGET ASMCALL BOMBR ACALL 30 SWAIT TARGET
2|ASMCALL 10 SWAIT BOMBADIER ASMCALL 60 SWAIT
3|REENTER AJMP
4|DATA LEFT2 ASM GAL2TBL SETPT LEFTROLL ACALL DIVE2 AJMP
5|DATA RIGHT2 ASM GAL2TBL SETPT RIGHTROLL ACALL DIVE2 AJMP
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      165-----
0|( ROLL GAL1 LEFT AND RIGHT )
1|DATA DIVE1 ASM TARGET ASMCALL BOMBR ACALL 10 SWAIT TARGET
2|ASMCALL 76 SWAIT REENTER AJMP
3|DATA LEFT1 ASM GAL1TBL SETPT LEFTROLL ACALL DIVE1 AJMP
4|DATA RIGHT1 ASM GAL1TBL SETPT RIGHTROLL ACALL DIVE1 AJMP
5|-->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      166-----
0|( RANDOM GORF GOODIES )
1|HEX
2|DATA GORFEXIT ASM 40 0 SETDC 11 SWAIT REENTER AJMP
3|DATA GALGORFR ASM 0 100 SETDC 0A AREPEAT GORF SETP 5 SWAIT
4|GORFB SETP 5 SWAIT ALOOP GORFEXIT AJMP
5|DATA GALGORF ASM 4800 SETXC NULPAT SETP
6|0 0 SETDC 0 0 SETDDC 28 SWAIT 0FE 0 SETS
7|RENTGAL SETR 1 SWAIT GORFB SETP 10 SWAIT
8|XADDWRITE SETR 1 GALGORFR RANDOMDO
9|0 -100 SETDC
10|0A AREPEAT GORF SETP 5 SWAIT GORFB SETP 5 SWAIT ALOOP
11|GORFEXIT AJMP
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      167-----
0|( LEFT PEELOFF FOR GALAXIAN 4 )
1|DATA DIVE4 ASM TARGET ASMCALL BOMBR ACALL 20 SWAIT TARGET
2|ASMCALL 40 SWAIT TARGET ASMCALL 46 SWAIT 3 GALGORF RANDOMDO
3|REENTER4 AJMP
4|DATA LEFT4 ASM GAL4TBL SETPT LEFTROLL ACALL DIVE4 AJMP
5|DATA RIGHT4 ASM GAL4TBL SETPT RIGHTROLL ACALL DIVE4 AJMP
6|-->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      168-----
0|( ATTACK PATH TABLES )
1|DECIMAL
2|DATA LEFTATBL LEFT1 , LEFT1 , LEFT2 ,
3|DATA RIGHTATBL RIGHT1 , RIGHT1 , RIGHT2 ,
4|DATA ATG1 32 B, 255 B, 11 B, 240 B, LEFT3 , 19 B, 0 B, LEFT3 ,
5|20 B, 0 B, LEFT4 , 255 B,
6|DATA ATG2 0 B, 144 B, 19 B, 0 B, RIGHT3 , 27 B, 16 B, RIGHT3 ,
7|20 B, 0 B, RIGHT4 , 255 B,
8|DATA ATG3 32 B, 255 B, 35 B, 240 B, LEFT3 , 43 B, 0 B, LEFT3 ,
9|44 B, 0 B, LEFT4 , 255 B,
10|DATA ATG4 0 B, 144 B, 43 B, 0 B, RIGHT3 , 51 B, 16 B, RIGHT3 ,
11|44 B, 0 B, RIGHT4 , 255 B,
12|DATA ATGTBL ATG1 , ATG2 , ATG3 , ATG4 ,
13|-->
14|
15|

```

```

+-----Block      169-----
0|( SUBROUTINE TO RESET THE ATTACK TIMER )
1|HEX SUBR SETATMR B PUSH, A C MOV, INVADERSLEFT LDA, 5 CPI,
2|CY~, IF, SKILLFACTOR LDA, A ANA,
3|0=, IF, LDAR, 3F ANI, ELSE, A DCR, 0=, IF, 0 C MVI, LDAR,
4|1F ANI, ELSE, 0 C MVI, A XRA, THEN, THEN,
5|A B MOV, INVADERSLEFT LDA, B ADD, C ADD, ATTACKTIMER STA,
6|THEN, B POP, RET,
7|( SUBROUTINE TO ABORT IF INVADER TOO CLOSE TO EDGES )
8|F= NOGO
9|SUBR CKPATH <ASSEMBLE H PUSH,
10|C A MOV, CALCINVT CALL, MASTERY LDED, D DAD, H A MOV,
11|H POP, 1E CPI, NOGO JRC, 094 CPI, NOGO JRNC,
12|M E MOV, H INX, M D MOV, XCHG, A ORA, RET,
13|LABEL NOGO A XRA, RET, ASSEMBLE>
14|DECIMAL -->
15|

```

```

+-----Block      170-----
0|( ATTACK ROUTINE FOR CODES 1 THRU 6 ) HEX
1|SUBR AT1T6
2|C A MOV, 4 CPI, CY, IF, LEFTINVN LDA, A DCR, LEFTATEL H LXI,
3|ELSE, RIGHTINVN LDA, 4 SUI, RIGHTATEL H LXI,
4|THEN, C ADD, A C MOV, 3 ANI,
5|RLC, A E MOV, 0 D MVI, D DAD,
6|H PUSH, XRACKBITS CALL, H POP, RZ,
7|CKPATH CALL, RZ, 0 B MVI,
8|ATSTART CALL, 10 A MVI, SETATMR JMP,
9|DECIMAL -->

```

```

10|
11|
12|
13|
14|
15|

```

```

+-----Block      171-----
0|( ATTACK ROUTINE FOR CODES 7-10 )
1|HEX F= ATSL F= PTL F= NOPE
2|SUBR ATG7T10 <ASSEMBLE
3|C A MOV, RLC, A C MOV, 0 B MVI, ATGTBL H LXI, B DAD,
4|M E MOV, H INX, M D MOV, XCHG, MASTERY 1 + LDA, M CMP,
5|RC, H INX, M CMP, RNC, H INX, H PUSH, 0 B MVI,
6|LABEL PTL M C MOV, H PUSH, XALIVEBITS CALL, 0<>, IF,
7|XRACKBITS CALL, 0<>, IF, B INR, ELSE, H POP, H POP, RET,
8|THEN, THEN, H POP, H INX, H INX, H INX, H INX, M A MOV, A INR,
9|PTL JRNZ, H POP, B ORA, RZ,
10|50 A MVI, SETATMR CALL,
11|LABEL ATSL M C MOV, H INX, M B MOV, H INX, M E MOV, H INX,
12|M D MOV, H INX,
13|C A MOV, A INR, RZ, H PUSH, D PUSH, B PUSH, XRACKBITS CALL,
14|B POP, H POP, NOPE JRZ, ATSTART CALL,
15|LABEL NOPE H POP, ATSL JMPR, ASSEMBLE> DECIMAL -->
+-----Block      172-----
0|( CHECK FOR ATTACK ROUTINE ) HEX
1|F= NOAT
2|CODE CHECKATTACK <ASSEMBLE X PUSHX, Y PUSHX, EXX,
3|ATTACKTIMER LHLD, H A MOV, L ORA, NOAT JRNZ,
4| LDAR, 0F ANI, A INR,
5|0D CPI, CY, IF, RRC, 7 ANI, A C MOV, AT1T6 CALL,
6| ELSE, 0D SUI, A C MOV, ATG7T10 CALL, THEN,
7|LABEL NOAT EXX, Y POPX, X POPX, NEXT
8|ASSEMBLE>
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      173-----
0|( PHASOR INTERCEPT CHECK ROUTINE )
1|F= INTLOG
2|SUBR PINTER <ASSEMBLE
3|PINTERFLAG LDA, A ANA, RNZ,
4|1 C MVI, CHECKALL CALL, 0<>, IF,
5|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
6|VYL Y L LDX, VYH Y H LDX, PINTERX SHLD,
7|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
8|VRACK Y C LDX, 6 C JCT, 0=, IF, XALIVEBITS CALL, M XRA,
9|A M MOV, THEN, C A YVI, INTLOG JMPR,
10|THEN, RACKCHECK CALL, RZ, 2 A MVI,
11|LABEL INTLOG PINTERFLAG STA, C A MOV, PINTERN STA,
12|verase CALL, PQSRH PQS X RESX,
13|RET, ASSEMBLE>
14|-->
15|

```

```

+-----Block      174-----
0|( GALAXIAN COLORS AND WAIT ROUTINE )
1|HEX
2|DATA GALCOLORS 7 B, 7D B, 0B B, 5A B, 7 B, 7D B, 0B B, 5A B,
3|
4|( WAIT FOR ATTACK TO END ROUTINE )
5|
6|: RACKWAIT 1 8 0 DO I RACKBITS B@ I ALIVEBITS B@
7|<> IF DROP 0 THEN LOOP ;
8|: WAITOUTATTACK BEGIN BMS RACKWAIT END SHUTUP ;
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      175-----
0|( INITIALIZE GALAXIAN GAME )
1|HEX : INITGAL 0 FLOOD INITMISSIONRAM 32 MISSION !
2|RESETRACK MAKEPATS DRAWMISSIONSCREEN
3|100 5000 408 A" GALAXIANS" COUNT SPOST
4|GALBUMPER BUMPMASSTERROUTINE ! 0 GALAXPAT INVPATAB !
5|GALAXNORMLPAT NORMLP1 ! 3000 MASTERX ! PINTER PHASINTR !
6|80 0 DO MASTERY @ I ANIMSTATE ! MASTERX @ I 1+ ANIMSTATE !
7|2 +LOOP ' WAITOUTATTACK REINIT ! 8 0 DO 0 I RACKBITS B! LOOP
8|7 0 ALIVEBITS B! 0F 1 ALIVEBITS B! 1F 2 ALIVEBITS B!
9|0F 3 ALIVEBITS B! 0F 4 ALIVEBITS B! 1F 5 ALIVEBITS B!
10|0F 6 ALIVEBITS B! 7 7 ALIVEBITS B!
11|20 INVADERSLEFT ! 0 LEFTINVN ! 38 RIGHTINVN !
12|0 PINTERFLAG ! BATOTAL 0 DO 0 I BOMBARRAY B! LOOP
13|GALFBA FBANIM ! ACTFB
14|GETNODE DUP PV1 ! 0 SWAP ! INITBOMBS 0 A2 VSTART
15|5 GALCOLORS FUC ; DECIMAL -->
+-----Block      176-----
0|( SCAN LOOP AND WAIT ROUTINE )
1|: GALSCAN WRTINV CHECKATTACK FIRECHECK PHASORINTERCEPTCHECK
2|PLAYERHITCHECK BMS ;
3|: GSWAIT WTIMER ! BEGIN WRTINV FIRECHECK PHASORINTERCEPTCHECK
4|BMS WTIMER @ 0 = END ;
5|: GSWAIT1 WTIMER ! BEGIN FIRECHECK PHASORINTERCEPTCHECK
6|BMS WTIMER @ 0 = END ;
7|DECIMAL
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      177-----
0|( ANIMATION STUFF TO DUMP OUT GALAXIANS )
1|DATA DUMPREENTER ASM 19200 SETXC NULPAT SETP RENTGAL SETR
2|1 SWAIT 0 PATI 20 SWAIT FLIPOVER ACALL 120 SWAIT AHALT
3|DATA DUMPGAL1 ASM GAL1TBL SETPT DUMPREENTER AJMP
4|DATA DUMPGAL2 ASM GAL2TBL SETPT DUMPREENTER AJMP
5|DATA DUMPGAL3 ASM GAL3TBL SETPT DUMPREENTER AJMP
6|DATA DUMPGAL4 ASM GAL4TBL SETPT 19200 SETXC NULPAT SETP
7|RENTGAL SETR 1 SWAIT 0 PATI 4 SWAIT FLIPOVER ACALL 120 SWAIT
8|AHALT
9|-->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      178-----
0|( DUMPOUT ROUTINE )
1|HEX 1A2 C= DUMPST DECIMAL
2|: PLYGA GASCORE B2MUSIC ;
3|: DUMPGALS EMUSIC E2MUSIC PLYGA WRTINV
4|57 0 DO DUMPGAL1 I DUMPST VSTART 8 +LOOP 120 GSWAIT1
5|PLYGA 58 1 DO DUMPGAL1 I DUMPST VSTART 8 +LOOP 110 GSWAIT
6|PLYGA 59 2 DO DUMPGAL2 I DUMPST VSTART 8 +LOOP 100 GSWAIT
7|PLYGA 52 11 DO DUMPGAL3 I DUMPST VSTART 8 +LOOP 100 GSWAIT
8|PLYGA DUMPGAL4 20 DUMPST VSTART DUMPGAL4 44 DUMPST VSTART
9|180 ATTACKTIMER ! ;
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      179-----
0|( SCAN LOOP AND STARTUP )
1|HEX
2|: GALAXIANS INITGAL DUMPGALS BEGIN GALSCAN
3|ENDOFFRAME @ END GALCOLORS SC 3 FDB ;
4|HEX A5 GSAB U! ' GALAXIANS GSAB 1+ U!
5|: BEGINGAME STARTGAME SKILLFACTOR ! GSAB 1+ @ DOIT ;
6|DECIMAL
7|;S
8|
9|
10|
11|
12|
13|
14|
15|

```



```

+-----Block      196-----
0|( SYSTEM LOAD ROUTINE ) 16 BASE !
1|CODE I 6EDD , 00 B, 66DD , 01 B, E5 B, NEXT
2|CODE UNMAP 0AF B, 0F8D3 , 0F9D3 , 0FF3E , 0FAD3 , NEXT
3|HERE CONSTANT .eot ( end of TERSE )
4| 0 VARIABLE .o ( #blks .eot - 4000 ) 0 VARIABLE .dp
5| 0 VARIABLE .t ( #blks 4000 - 8000 ) 0 VARIABLE .vp
6| 0 VARIABLE .h ( #blks 8000 - HERE ) 0 VARIABLE .la
7| 1 VARIABLE .f ( #blks F000 - FFFF )
8|: bload ( from-blk to-addr #blks --- next-blk )
9| DUP >R 0 DO 2DUP DROP I + BLOCK 2DUP DROP
10| I 400 * + 400 UNPROT BMOVE PROT LOOP DROP R> + ;
11|: boot .o 1 bload .eot .o @ bload
12| 4000 .t @ bload 8000 .h @ bload F000 .f @ bload
13| .dp @ DP ! .vp @ VPTR ! .la @ LAST ! ;
14|UNMAP SCR @ 1+ boot DECIMAL ." 03-18-80" . fast OK ;S
15|0A BASE ! ;S

```

```

+-----Block      198-----
0|( SYSTEM LOAD ROUTINE ) 16 BASE !
1|CODE I 6EDD , 00 B, 66DD , 01 B, E5 B, NEXT
2|CODE UNMAP 0AF B, 0F8D3 , 0F9D3 , 0FF3E , 0FAD3 , NEXT
3|HERE CONSTANT .eot ( end of TERSE )
4| 0 VARIABLE .o ( #blks .eot - 4000 ) 0 VARIABLE .dp
5| 0 VARIABLE .t ( #blks 4000 - 8000 ) 0 VARIABLE .vp
6| 0 VARIABLE .h ( #blks 8000 - HERE ) 0 VARIABLE .la
7| 1 VARIABLE .f ( #blks F000 - FFFF )
8|: bload ( from-blk to-addr #blks --- next-blk )
9| DUP >R 0 DO 2DUP DROP I + BLOCK 2DUP DROP
10| I 400 * + 400 UNPROT BMOVE PROT LOOP DROP R> + ;
11|: boot .o 1 bload .eot .o @ bload
12| 4000 .t @ bload 8000 .h @ bload F000 .f @ bload
13| .dp @ DP ! .vp @ VPTR ! .la @ LAST ! ;
14|UNMAP SCR @ 1+ boot DECIMAL ." 03-18-80" . fast OK ;S
15|0A BASE ! ;S

```

```

+-----Block      199-----
0|( SYSTEM LOAD ROUTINE ) 16 BASE !
1|CODE I 6EDD , 00 B, 66DD , 01 B, E5 B, NEXT
2|CODE UNMAP 0AF B, 0F8D3 , 0F9D3 , 0FF3E , 0FAD3 , NEXT
3|HERE CONSTANT .eot ( end of TERSE )
4| 0 VARIABLE .o ( #blks .eot - 4000 ) 0 VARIABLE .dp
5| 0 VARIABLE .t ( #blks 4000 - 8000 ) 0 VARIABLE .vp
6| 0 VARIABLE .h ( #blks 8000 - HERE ) 0 VARIABLE .la
7| 1 VARIABLE .f ( #blks F000 - FFFF )
8|: bload ( from-blk to-addr #blks --- next-blk )
9| DUP >R 0 DO 2DUP DROP I + BLOCK 2DUP DROP
10| I 400 * + 400 UNPROT BMOVE PROT LOOP DROP R> + ;
11|: boot .o 1 bload .eot .o @ bload
12| 4000 .t @ bload 8000 .h @ bload F000 .f @ bload
13| .dp @ DP ! .vp @ VPTR ! .la @ LAST ! ;
14|UNMAP SCR @ 1+ boot DECIMAL ." 03-18-80" . fast OK ;S
15|0A BASE ! ;S

```

100|(BEGINING OF ATTACK FIGHTER GAME)
101|(ATTACK FIGHTER PATTERNS - LEADER PATTERN)
102|(MISSIONS- LASAR LZ) HEX
150|(ATF VARIABLES)
151|(PHASOR INTERCEPT CHECK ROUTINE)
152|(TIME BASED VECTOR UPDATE - WITH LIMIT CHECKING)
153|(FORMATION LEADERS ALMOST NULL INTERRUPT ROUTINE)
154|(ANIMATION TO ACTIVATE FORMATIONS)
155|(ROUTINE TO ACTIVATE THE FORMATIONS)
156|(KAMIKAZE ATTACK COORDINATOR)
157|(FORMATION MOVE ROUTINE - RANDOM MOVES FIGHTER FORMATIONS)
158|(INTERRUPT ROUTINE TO DRAW LASER BLAST)
159|(LASER ANIMATION AND VECTOR START ROUTINE)
160|(CHECK FORMATION STATE VARIABLE AND EITHER FIRE OR REVECTOR)
161|(FORMATION MOVE CHECK ROUTINE)
162|(ANIMATION LIST FOR FIREBASE STUFF)
163|(ATTACK FIGHTERS COLORS AND WAIT ROUTINE)
164|(INITIALIZE ATTACK FIGHTERS GAME)
165|(SCAN LOOP AND STARTUP)

```

+-----Block      100-----
0|( BEGINING OF ATTACK FIGHTER GAME )
1|DATA GSAB 0 B, 0 ,
2|-->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      101-----
0|( ATTACK FIGHTER PATTERNS - LEADER PATTERN )
1|DATA LEADER 4 B, 11 B, QUAD
2|0000 B, 0000 B, 3000 B, 0000 B,
3|0003 B, 3333 B, 3000 B, 0000 B,
4|0000 B, 0220 B, 2000 B, 0000 B,
5|0000 B, 0220 B, 0000 B, 0000 B,
6|0000 B, 2220 B, 0220 B, 0000 B,
7|1111 B, 2222 B, 2220 B, 0000 B,
8|0000 B, 2220 B, 0220 B, 0000 B,
9|0000 B, 0220 B, 0000 B, 0000 B,
10|0000 B, 0220 B, 2000 B, 0000 B,
11|0003 B, 3333 B, 3000 B, 0000 B,
12|0000 B, 0000 B, 3000 B, 0000 B,
13|DECIMAL -->
14|
15|

+-----Block      102-----
0|( MISSIONS- LASAR LZ ) HEX
1|DATA LZSCORE ASM
2| 28 MASTER #G2 #D3 #A4 TONES CC ABVOLS 1C MCVOLS
3| 0 1 1 20 MOVENOISE 1 2 0 MOVESOUND 1 COUNTPANS PLAY
4| 20 1 -1 0 MOVENOISE 1 -1 28 8 RAMBLE 1 COUNTPANS PLAY
5| KBSCORE LDPCC ( jump to background sound )
6|DECIMAL ;S
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      150-----
0|( ATF VARIABLES )
1|0 V= TBV1 0 V= TBV2 5 ARRAY F1 5 ARRAY F2
2|0 V= FSV1 0 V= FSV2
3|DECIMAL -->
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      151-----
0|( PHASOR INTERCEPT CHECK ROUTINE )
1|DECIMAL F= INTLOG
2|SUBR PINTER <ASSEMBLE
3|PINTERFLAG LDA, A ANA, RNZ,
4|1 C MVI, CHECKALL CALL, RZ,
5|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
6|VYL Y L LDX, VYH Y H LDX, PINTERY SHLD,
7|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
8|VRACK Y A LDX, PINTERN STA,
9|1 A MVI,
10|PINTERFLAG STA,
11|( INVADERSLEFT LDA, A DCR, INVADERSLEFT STA, )
12|verase CALL, PQSRH PQS X RESX,
13|RET, ASSEMBLE> -->
14|
15|

+-----Block      152-----
0|( TIME BASED VECTOR UPDATE - WITH LIMIT CHECKING )
1|DECIMAL F= LCD1 F= LCD2
2|SUBR VUPDLC <ASSEMBLE
3|C A MOV, A ANA, RZ, ( DONT IF ZERO VECTORING WANTED )
4|VXL X L LDX, VXH X H LDX, VDXL X E LDX, VDXH X D LDX, C B MOV,
5|LABEL LCD1 D DAD, LCD1 DJNZ, H A MOV, VDDXL X CMPX, CY, IF,
6|VDDXL X H LDX, 0 L MVI, L VDXL X STX, L VDXH X STX, ELSE,
7|VDDXH X CMPX, CY~, IF, VDDXH X H LDX, 0 L MVI, L VDXL X STX,
8|L VDXH X STX, THEN, THEN, L VXL X STX, H VXH X STX,
9|VYL X L LDX, VYH X H LDX, VDYL X E LDX, VDYH X D LDX, C B MOV,
10|LABEL LCD2 D DAD, LCD2 DJNZ, H A MOV, VDDYL X CMPX, CY, IF,
11|VDDYL X H LDX, 0 L MVI, L VDYL X STX, L VDYH X STX, ELSE,
12|VDDYH X CMPX, CY~, IF, VDDYH X H LDX, 0 L MVI, L VDYL X STX,
13|L VDYH X STX, THEN, THEN, L VYL X STX, H VYH X STX,
14|40 VXZW X MVIX, RET, ASSEMBLE>
15|DECIMAL -->

```

```

+-----Block      153-----
0| ( FORMATION LEADERS ALMOST NULL INTERRUPT ROUTINE )
1| SUBR FLEADER TBCALC CALL, VUPDLC CALL, aup CALL, KILLOFF JMP,
2| DECIMAL -->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      154-----
0| ( ANIMATION TO ACTIVATE FORMATIONS )
1| HEX
2| DATA TBVTL ASM FLEADER SETR NULPAT SETP 4010 0A00C SETDDC
3| FOREVER 120 SWAIT EVERFOR
4| DATA ATBV1 ASM 3800 SETXC 1000 SETYC TBVTL AJMP
5| DATA ATBV2 ASM 3800 SETXC 4800 SETYC TBVTL AJMP
6| DATA ALEADER ASM LEADER SETP FOREVER 120 SWAIT EVERFOR
7| DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      155-----
0| ( ROUTINE TO ACTIVATE THE FORMATIONS )
1| HEX : STARTFORMATIONS GETNODE TBV1 ! GETNODE TBV2 !
2| 5 0 DO GETNODE I F1 ! GETNODE I F2 ! LOOP
3| ATBV1 @ 0BA TBV1 @ XVSTART ATBV2 @ 0BA TBV2 @ XVSTART
4| TBV1 @ 400 0 AKAMI 03 1B2 0 F1 @ FSTART
5| TBV2 @ 400 0 AKAMI 03 1B2 0 F2 @ FSTART
6| TBV1 @ 400 1000 AKAMI 03 1B2 1 F1 @ FSTART
7| TBV2 @ 400 1000 AKAMI 03 1B2 1 F2 @ FSTART
8| TBV1 @ 400 2000 AKAMI 03 1B2 2 F1 @ FSTART
9| TBV2 @ 400 2000 AKAMI 03 1B2 2 F2 @ FSTART
10| SKILLFACTOR @ IF TBV1 @ 800 1000 AKGORF 03 1B2 3 F1 @ FSTART
11| TBV2 @ 800 1000 AKGORF 03 1B2 3 F2 @ FSTART 0A ELSE 8 THEN
12| INVADERSLEFT ! TBV1 @ 0 1000 ALEADER 04 1B2 4 F1 @ FSTART
13| TBV2 @ 0 1000 ALEADER 04 1B2 4 F2 @ FSTART ;
14| DECIMAL -->
15|

```

```

+-----Block      156-----
0|( KAMIKAZE ATTACK COORDINATOR )
1|HEX SUBR KAMIATC ATTACKTIMER LDA, A ANA, RNZ,
2|LDAR, 7 ANI, 4 CPI, CY, IF, 0 F1 H LXI,
3|ELSE, 0 F2 H LXI, 3 ANI, THEN, RLC, A E MOV, 0 D MVI,
4|D DAD, M E MOV, H INX, M D MOV, D PUSH, X POPX, DI,
5|PQSRH PQS X BITX, RZ, ASFLOK VAUXS X BITX, RZ,
6|VYH X A LDX, 20 SUI, 90 CPI, RNC,
7|LDAR, 1 ANI, 0=, IF, KAMIATL H LXI, ELSE, KAMIATR H LXI,
8|THEN, ASFLOK VAUXS X RESX, CRASHA CALL, LDAR, 7F ANI,
9|20 ADI, ATTACKTIMER STA, PLAYKBS JMP,
10|
11|CODE CKKAMI X PUSHX, Y PUSHX, B PUSH, KAMIATC CALL, EI,
12|B POP, Y POPX, X POPX, NEXT
13|DECIMAL -->
14|
15|
+-----Block      157-----
0|( FORMATION MOVE ROUTINE - RANDOM MOVES FIGHTER FORMATIONS )
1|HEX SUBR FMOVER ( IN IX=FORM VECT DE=Y BIAS )
2|D PUSH, SKILLFACTOR LDA, A ANA, 0=, IF, 40 D LXI, D PUSH,
3|30 D LXI, ELSE, 20 D LXI, D PUSH, 18 D LXI, THEN,
4|rnd CALL, D POP, D DAD, H PUSH, ( TIME )
5|2000 D LXI, rnd CALL, 2000 D LXI, D DAD, D POP, D PUSH, DI,
6|VXL X C LDX, VXH X B LDX, CDELTA CALL, L VXL X STX,
7|H VXH X STX, E VDXL X STX, D VDXH X STX,
8|4000 D LXI, rnd CALL, D POP, B POP, D PUSH, B DAD,
9|VYL X C LDX, VYH X B LDX, CDELTA CALL,
10|L VYL X STX, H VYH X STX, E VDYL X STX, D VDYH X STX, EI,
11|D POP, RET,
12|DECIMAL -->
13|
14|
15|
+-----Block      158-----
0|( INTERRUPT ROUTINE TO DRAW LASER BLAST )
1|( VDDXL=STATE VAR, VDDXH=X COUNTER, VDDYHL=SCREEN ADDR )
2|SUBR BUMPLAZ A INR, A VDDXL X STX, VXH X A LDX, A VDDXH X STX,
3|VSAL X L LDX, VSAH X H LDX, L VDDYL X STX, H VDDYH X STX, RET,
4|HEX F= DRL
5|SUBR SLASER <ASSEMBLE PQT B X C LDX, 0 PQT B X MVIX,
6|C A MOV, A ANA, KILLOFF JZ,
7|VDDXL X A LDX, A ANA, 0=, IF, BUMPLAZ CALL, THEN,
8|VDDXH X A LDX, A ANA, 0=, IF, VDDXL X A LDX, 2 CPI,
9|0=, IF, PQSRH PQS X RESX, ELSE, BUMPLAZ CALL, THEN,
10|ELSE, C B MOV, C SUB, 0<, IF, C ADD, A B MOV, THEN,
11|VDDXH X A LDX, B SUB, A VDDXH X STX, 20 A MVI, YAGIC OUT,
12|VDDYL X L LDX, VDDYH X H LDX,
13|LABEL DRL H DCX, ED X MVI, DRL DJNZ,
14|L VDDYL X STX, H VDDYH X STX,
15|THEN, KILLOFF JMP, ASSEMBLE> DECIMAL -->

```

```

+-----Block      159-----
0|( LASER ANIMATION AND VECTOR START ROUTINE )
1|HEX
2|DATA LASERA ASM SLASER SETR NULPAT SETP 4 SWAIT
3|
4|SUBR LSHOT DI, VXL Y L LDX, VXH Y H LDX, H PUSH,
5|VYL Y L LDX, VYH Y H LDX, 0500 D LXI, D DAD, H PUSH,
6|LASERA H LXI, H PUSH,
7|0 H LXI, H PUSH,
8|0A2 H LXI, H PUSH,
9|XYVSTART JMP,
10|DECIMAL -->
11|
12|
13|
14|
15|

+-----Block      160-----
0|( CHECK FORMATION STATE VARIABLE AND EITHER FIRE OR REVECTOR )
1|SUBR ZAPFORM ( FREEZE VECTOR POINTED AT BY IX )
2|A XRA, A VDXL X STX, A VDXH X STX, A VDYL X STX, A VDYH X STX,
3|RET,
4|
5|SUBR FCHECK M A MOV, A ANA, 0=, IF,
6|A INR, A M MOV, ( LASER SHOOTER ) DI,
7|PQSRH PQS Y BITX, 0<>, IF, ZAPFORM CALL,
8|VXH X A LDX, A INR, A E MOV, 0 D MVI, D PUSH, ( TIME STUFF )
9|LSHOT CALL, LZSCORE H LXI, MB2 Y LXIX, pmusic CALL, D POP,
10|THEN, ELSE, A XRA, A M MOV, FMOVER CALL,
11|THEN, RET,
12|DECIMAL -->
13|
14|
15|

+-----Block      161-----
0|( FORMATION MOVE CHECK ROUTINE )
1|F= NC1 F= NC2
2|HEX CODE FMC <ASSEMBLE
3|X PUSHX, Y PUSHX, B PUSH,
4|TIMER1 LDA, A ANA, NC1 JRNZ, 4 F1 LIYD,
5|TBV1 LIXD, 1000 D LXI, FSV1 H LXI, FCHECK CALL, TIMER1 SDSD,
6|LABEL NC1 TIMER3 LDA, A ANA, NC2 JRNZ, 4 F2 LIYD,
7|TBV2 LIXD, 4000 D LXI, FSV2 H LXI, FCHECK CALL, TIMER3 SDSD,
8|LABEL NC2
9|B POP, Y POPX, X POPX, NEXT ASSEMBLE>
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      162-----
0|( ANIMATION LIST FOR FIREBASE STUFF )
1|SUBR ATFINTER CKATRS CALL, ( 0<>, IF, INVADERSLEFT LDA, A DCR,
2|INVADERSLEFT STA, THEN, ) EXPLODEFB CALL,
3|X PUSHX, TBV1 LIXD, ZAPFORM CALL,
4|TBV2 LIXD, ZAPFORM CALL, X POPX, RET,
5|HEX DATA ATFFBA ASM ATFINTER SETI 2005 B005 SETDDC PLAYERANIM
6|AJMP DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      163-----
0|( ATTACK FIGHTERS COLORS AND WAIT ROUTINE )
1|HEX
2|DATA ATFCOLORS 7 B, 7D B, 0B B, 5A B, 7 B, 7D B, 0B B, 5A B,
3|
4|F= NYD F= YWD F= SAL
5|CODE SCANARRAY <ASSEMBLE EXX, H POP, X PUSHX, 5 B MVI,
6|LABEL SAL M E MOV, H INX, M D MOV, H INX,
7|D PUSH, X POPX, PQSRH PQS X BITX,
8|0<>, IF, ASFLOK VAUXS X BITX, NYD JRZ, THEN,
9|SAL DJNZ, 1 H LXI, YWD JMPR,
10|LABEL NYD 0 H LXI,
11|LABEL YWD X POPX, H PUSH, EXX, NEXT ASSEMBLE>
12|: ATFWAIT BEGIN BMS 0 F1 SCANARRAY 0 F2 SCANARRAY AND END
13|SHUTUP ; DECIMAL -->
14|
15|

```

```

+-----Block      164-----
0|( INITIALIZE ATTACK FIGHTERS GAME )
1|HEX : INITATF 0 FLOOD INITMISSIONRAM 33 MISSION !
2|DRAWMISSIONSCREEN
3|100 5000 408 A" ATTACK FIGHTERS" COUNT SPOST
4|0 PINTERFLAG ! PINTER PHASINTR ! ' ATFWAIT REINIT !
5|1 FSV1 ! 1 FSV2 !
6|ATFFBA FBANIM ! ACTFB
7|GETNODE DUP PV1 ! 0 SWAP !
8|38 ATTACKTIMER ! 10 TIMERS ! 48 TIMERS ! ;
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```


+-----Block 165-----

0| (SCAN LOOP AND STARTUP)

1| : ATFSCAN FIRECHECK PHASORINTERCEPTCHECK CKKAMI FMC

2| BMS PLAYERHITCHECK ;

3| HEX : ATF INITATF STARTFORMATIONS 5 ATFCOLORS FUC

4| EMUSIC E2MUSIC

5| BEGIN ATFSCAN ENDOFFRAME @ END

6| 5 FDB ;

7| HEX A5 GSAB UI ' ATF GSAB 1+ UI

8| : BEGINGAME STARTGAME SKILLFACTOR ! GSAB 1+ @ DOIT ;

9| DECIMAL ;S

10|

11|

12|

13|

14|

15|

```

100|C BEGIN MISSION 4 GOODIES )
101|C TIE FIGHTER 1 )
102|C TIE FIGHTER PATTERN 2 )
103|C TIE FIGHTER PATTERNS 3 AND 4 )
104|C X WING PATTERN 1 )
105|C X WING PATTERNS 2 AND 5 )
106|C X WING PATTERNS 3 AND 4 )
107|C PHOTON TORPEDO PATTERNS )
108|C PHOTON TORPEDO PATTERNS CONTINUED )
109|C FINAL PHOTON TORPEDO DISINTEGRATION PATTERN )
110|C SHIP SPIRAL- SP, FIREBLAST FB ) HEX
111|C MISSIONS- STAR SPIRAL SOUND ) HEX
112|C MISSIONS- BLACK HOLE EMERGENCE ) HEX
150|C MISSION FOUR - SPACE WARP )
151|C PHASOR INTERCEPT CHECK ROUTINE )
152|C CHECK WITH INTERCEPT WITH FIREBASE - IF SO KILL IT )
153|C CHECK FOR ATTACKER - FIREBASE INTERCEPT )
154|C COROUTINE GOODIES )
155|C POINT WRITE ROUTINE STUFF )
156|C POSITION TABLE FOR ATTACKERS LEFT DISPLAY )
157|C SPIRAL ANIMATION SUBROUTINES )
158|C ANIMATION SEQUENCES TO START SPIRALING ATTACKERS )
159|C RETURN SPIRAL STATUS BASED ON SKILL FACTOR )
160|C COMMAND TO START SPIRALING ATTACKERS )
161|C ANIMATION FOR PHOTON TORPEDO ATTACK )
162|C CHECK PHOTON TORPEDOS )
163|C COLOR TABLE -- WAIT FOR ATTACK TO END )
164|C LINE EFFECT COROUTINE )
165|C INITIALIZE MISSION 4 - DOGFIGHT IN THE SPACE WARP )
166|C SCAN LOOP AND STARTUP )

```

```
+-----Block    100-----
0|( BEGIN MISSION 4 GOODIES )
1|DATA GSAB 0 B, 0 ,
2|-->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
```

```
+-----Block    101-----
0|( TIE FIGHTER 1 )
1|DATA TF1 3 B, 11 B, QUAD
2|~ 1111 1111 1000 ^
3|~ 0000 2000 0000 ^
4|~ 0100 2001 0000 ^
5|~ 0002 2200 0000 ^
6|~ 0022 3220 0000 ^
7|~ 0022 3220 0000 ^
8|~ 0022 3220 0000 ^
9|~ 0002 2200 0000 ^
10|~ 0100 2001 0000 ^
11|~ 0000 2000 0000 ^
12|~ 1111 1111 1000 ^
13|DECIMAL -->
14|
15|
```

```
+-----Block    102-----
0|( TIE FIGHTER PATTERN 2 )
1|DATA TF2 3 B, 9 B, QUAD
2|~ 1111 1110 0000 ^
3|~ 0002 0000 0000 ^
4|~ 1022 2010 0000 ^
5|~ 0023 2000 0000 ^
6|~ 0223 2200 0000 ^
7|~ 0023 2000 0000 ^
8|~ 1022 2010 0000 ^
9|~ 0002 0000 0000 ^
10|~ 1111 1110 0020 ^
11|DECIMAL -->
12|
13|
14|
15|
```

```

+-----Block      103-----
0|( TIE FIGHTER PATTERNS 3 AND 4 )
1|DATA TF3 2 B, 7 B, QUAD
2|~ 1111 1000 ^
3|~ 0020 0000 ^
4|~ 0222 0000 ^
5|~ 0232 0000 ^
6|~ 0222 0000 ^
7|~ 0020 0000 ^
8|~ 1111 1000 ^
9|DECIMAL DATA TF4 2 B, 4 B, QUAD
10|~ 1110 0000 ^
11|~ 0200 0000 ^
12|~ 0200 0000 ^
13|~ 1110 0000 ^
14|DECIMAL -->
15|

```

```

+-----Block      104-----
0|( X WING PATTERN 1 )
1|DATA XW1 4 B, 11 B, QUAD
2|~ 2200 0000 0220 0000 ^
3|~ 2200 0000 0220 0000 ^
4|~ 0010 0100 1000 0000 ^
5|~ 0001 0101 0000 0000 ^
6|~ 0001 1111 0000 0000 ^
7|~ 2011 3331 1020 0000 ^
8|~ 0001 1111 0000 0000 ^
9|~ 0001 0101 0000 0000 ^
10|~ 0010 0100 1000 0000 ^
11|~ 2200 0000 0220 0000 ^
12|~ 2200 0000 0220 0000 ^
13|DECIMAL -->
14|
15|

```

```

+-----Block      105-----
0|( X WING PATTERNS 2 AND 5 )
1|DATA XW2 3 B, 5 B, QUAD
2|~ 2200 0002 2000 ^
3|~ 2200 0002 2000 ^
4|~ 0010 1010 0000 ^
5|~ 0001 1100 0000 ^
6|~ 2011 3110 2000 ^
7|~ 0001 1100 0000 ^
8|~ 0010 1010 0000 ^
9|~ 2200 0002 2000 ^
10|~ 2200 0002 2000 ^
11|DECIMAL DATA XW3 2 B, 3 B, QUAD
12|~ 2020 0000 ^
13|~ 0100 0000 ^
14|~ 2020 0000 ^
15|DECIMAL -->

```

```

+-----Block      106-----
0|( X WING PATTERNS 3 AND 4 )
1|DATA XW3 3 B, 7 B, QUAD
2|~ 2000 0020 0000 ^
3|~ 0101 0100 0000 ^
4|~ 0011 1000 0000 ^
5|~ 2113 1120 0000 ^
6|~ 0011 1000 0000 ^
7|~ 0101 0100 0000 ^
8|~ 2000 0020 0030 ^
9|DECIMAL DATA XW4 2 B, 5 B, QUAD
10|~ 2000 2000 ^
11|~ 0111 0000 ^
12|~ 2131 2000 ^
13|~ 0111 0000 ^
14|~ 2000 2000 ^
15|DECIMAL -->

```

```

+-----Block      107-----
0|( PHOTON TORPEDO PATTERNS )
1|DATA PT1 1 B, 1 B, QUAD
2|1000 B,
3|DECIMAL DATA PT2 2 B, 3 B, QUAD
4|~ 0300 0000 ^
5|~ 3130 0000 ^
6|~ 0300 0000 ^
7|DECIMAL DATA PT3 2 B, 4 B, QUAD
8|~ 0330 0000 ^
9|~ 3113 0000 ^
10|~ 3113 0000 ^
11|~ 0330 0000 ^
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      108-----
0|( PHOTON TORPEDO PATTERNS CONTINUED )
1|DATA PT4 2 B, 5 B, QUAD
2|~ 0333 0000 ^
3|~ 3111 3000 ^
4|~ 3111 3000 ^
5|~ 3111 3000 ^
6|~ 0333 0000 ^
7|DECIMAL DATA PT5 2 B, 5 B, QUAD
8|~ 0300 1000 ^
9|~ 3101 0000 ^
10|~ 0111 3000 ^
11|~ 0031 1000 ^
12|~ 3003 0000 ^
13|DECIMAL -->
14|
15|

```

```

+-----Block      109-----
0|( FINAL PHOTON TORPEDO DISINTEGRATION PATTERN )
1|DATA PT6 2 B, 6 B, QUAD
2|~ 0100 0000 ^
3|~ 0001 0000 ^
4|~ 0300 1000 ^
5|~ 1030 0000 ^
6|~ 0001 0000 ^
7|~ 3000 3000 ^
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|

+-----Block      110-----
0|( SHIP SPIRAL- SP, FIREBLAST FB ) HEX
1| DATA SPSCORE ASM 57 MASTER 1 -5 57 0C RAMBLE 1 COUNTLIMITS
2| 1 -2 3F MOVESOUND
3| 99 ABVOLS 1A MCVOLS #C3 #E3 #FS3 TONES PLAY QUIET
4|: SP E2MUSIC SPSCORE B2MUSIC ;
5|DATA FBLSCORE ASM #E1 #F2 #G2 TONES 1 -2 3F MOVESOUND
6| 40 MASTER 1 -2 A0 20 RAMBLE 2 COUNTLIMITS 20 NOISE
7| 99 ABVOLS 1A MCVOLS PLAY QUIET
8|: FBL E2MUSIC FBLSCORE B2MUSIC ;
9|-->
10|
11|
12|
13|
14|
15|

+-----Block      111-----
0|( MISSIONS- STAR SPRIAL SOUND ) HEX
1|DATA ST1SCORE ASM
2| #E2 #B2 #FS3 TONES 1 1 0 MOVESOUND 0C NOISE
3| 40 MASTER 2 -1 40 10 RAMBLE 1 COUNTLIMITS
4| 88 ABVOLS 18 MCVOLS PLAY 1 -1 1 MOVETB RERAMBLE 2 COUNTLIMITS
5| PLAY 1 F2 2 MOVEHIGHLIM 1 1 2 MOVESTEP 0 NOISE
6| RERAMBLE 2 COUNTLIMITS PLAY 1 2 4 MOVESTEP 6 COUNTLIMITS
7| RERAMBLE PLAY QUIET
8|DATA ST2SCORE ASM
9| #G1 #D2 #A2 TONES 1 -1 3F MOVESOUND 0C NOISE
10| 40 MASTER 2 -1 40 10 RAMBLE 1 COUNTLIMITS
11| 88 ABVOLS 10 MCVOLS PLAY 1 -1 1 MOVETB RERAMBLE 2 COUNTLIMITS
12| PLAY 1 F2 2 MOVEHIGHLIM 1 1 2 MOVESTEP 0 NOISE
13| RERAMBLE 2 COUNTLIMITS PLAY 1 2 4 MOVESTEP 6 COUNTLIMITS
14| RERAMBLE PLAY QUIET
15|: ST ST1SCORE PMUSIC ST2SCORE P2MUSIC ; DECIMAL ;S

```

```

+-----Block      112-----
0|( MISSIONS- BLACK HOLE EMERGENCE ) HEX
1|DATA BH1SCORE ASM
2| 11 40 62 TONES 10 MASTER 1 4 C0 10 RAMBLE 1 COUNTLIMITS
3| 2 1 0 MOVESOUND 0 1 4 B0 MOVENOISE 88 ABVOLS 1C MCVOLS PLAY
4| C0 MASTER 1 -8 C0 2 RAMP B0 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
5| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
6| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
7| 2 MASTER 3 1 FF 2 RAMBLE 1 COUNTLIMITS PLAY QUIET
8|DATA BH2SCORE ASM
9| 13 30 50 TONES 10 MASTER 1 4 C0 10 RAMBLE 1 COUNTLIMITS
10| 2 1 0 MOVESOUND 0 1 4 B0 MOVENOISE 88 ABVOLS 1C MCVOLS PLAY
11| C0 MASTER 1 -8 C0 2 RAMP B0 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
12| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
13| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
14| 2 MASTER 3 1 FF 2 RAMBLE 1 COUNTLIMITS PLAY QUIET
15|: BH BH1SCORE PMUSIC BH2SCORE P2MUSIC ; DECIMAL ;S
+-----Block      150-----
0|( MISSION FOUR - SPACE WARP )
1|TIMER3 C= LETIMER
2|0 V= AV ( ATTACKER VECTOR ADDRESS )
3|0 V= ATTACKERSLEFT ( NUMBER OF ATTACKERS FOR MISSION 4 )
4|-->
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      151-----
0|( PHASOR INTERCEPT CHECK ROUTINE )
1|DECIMAL F= INTLOG
2|SUBR M4PINTER <ASSEMBLE
3|PINTERFLAG LDA, A ANA, RNZ,
4|AV LIYD, CHECKVEC CALL, RZ,
5|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
6|VYL Y L LDX, VYH Y H LDX, PINTERY SHLD,
7|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
8|VRACK Y A LDX, PINTERX STA,
9|1 A MVI,
10|PINTERFLAG STA,
11|verase CALL, PQSRH PQS X RESX,
12|RET, ASSEMBLE> -->
13|
14|
15|

```

```

+-----Block      152-----
0|( CHECK WITH INTERCEPT WITH FIREBASE - IF SO KILL IT )
1|SUBR FBHCHECK 0 FBVECTOR Y LXIX, CHECKVEC CALL, RZ,
2|X PUSHX, 0 FBVECTOR X LXIX, EXPLODEFB CALL,
3|X POPX, PQSRH PQS X RESX, PQSDW PQS X SETX,
4|erase CALL,
5|1 A MVI, A ANA, RET,
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      153-----
0|( CHECK FOR ATTACKER - FIREBASE INTERCEPT )
1|DECIMAL
2|SUBR DIDIHITPLAYER
3|PINTERFLAG LDA, A ANA, RNZ,
4|FBHCHECK CALL, RZ,
5|VYL X L LDX, VYH X H LDX, PINTERY SHLD,
6|VXL X L LDX, VXH X H LDX, PINTERX SHLD,
7|VRACK X A LDX, PINTERN STA,
8|1 A MVI,
9|PINTERFLAG STA,
10|RET,
11|-->
12|
13|
14|
15|

```

```

+-----Block      154-----
0|( COROUTINE GOODIES )
1|0 V= LEPC
2|CODE LETCK LETIMER LHLD, H A MOV, L ORA,
3|0=, IF, LEPC LHLD, LEPC SB CD, L C MOV, H B MOV,
4|THEN, NEXT
5|
6|CODE LWAIT H POP, LETIMER SHLD, LEPC LHLD, LEPC SB CD,
7|L C MOV, H B MOV, NEXT
8|
9|: SETLEPC 1+ LEPC I ;
10|
11|DECIMAL -->
12|
13|
14|
15|

```



```

+-----Block      155-----
0|( POINT WRITE ROUTINE STUFF )
1|HEX
2|( SUBROUTINE TO DRAW A POINT )
3|SUBR UPPOINT DI, H A MOV, C0 CPI, RNC, D A MOV, 50 CPI, RNC,
4|  C A MOV, RRC, RRC, A B MOV, 20 C MVI, ( FUDGE B )
5| relabs CALL, C A MOV, MAGIC OUT, B M MOV, ( WRITE IT )
6|EI, RET,
7|
8|CODE POINT EXX, B POP, H POP, D POP, UPPOINT CALL, EXX, NEXT
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      156-----
0|( POSITION TABLE FOR ATTACKERS LEFT DISPLAY )
1|{ : XT } 64 * , { ; } { : YT } 256 * , { ; }
2|TABLE ATXPOS 206 XT 205 XT 203 XT 200 XT 197 XT 195 XT
3|194 XT 195 XT 197 XT 200 XT 203 XT 205 XT 203 XT 202 XT
4|200 XT 198 XT 197 XT 198 XT 200 XT 202 XT
5|TABLE ATYPOS 100 YT 103 YT 105 YT 106 YT 105 YT 103 YT 100 YT
6|97 YT 95 YT 94 YT 95 YT 97 YT 100 YT 102 YT 103 YT 102 YT
7|100 YT 98 YT 97 YT 98 YT
8|: SHOWATTACKERS ATTACKERSLEFT @ 0 DO
9|I ATXPOS @ I ATYPOS @ 2 POINT LOOP ;
10|DECIMAL -->
11|
12|
13|
14|
15|
+-----Block      157-----
0|( SPIRAL ANIMATION SUBROUTINES )
1|DECIMAL DATA GORFR ASM GORF1 SETP 90 SWAIT GORF2 SETP 80 SWAIT
2|GORF3 SETP 70 SETP GORF4 SETP 60 SWAIT GORF5 SETP 50 SWAIT
3|GORF SETP FOREVER 120 SWAIT EVERFOR
4|DATA XWF ASM XW5 SETP 110 SWAIT XW4 SETP 90 SWAIT
5|XW3 SETP 80 SWAIT XW2 SETP 70 SWAIT XW1 SETP
6|FOREVER 120 SWAIT EVERFOR
7|DATA TFF ASM
8|NULPAT SETP SPWRITE SETR DIDHITPLAYER SETI
9|I SWAIT 3 GORFR RANDOMDO 1 XWF RANDOMDO
10|TF4 SETP 120 SWAIT TFF SETP 110 SWAIT
11|TF2 SETP 90 SWAIT TF1 SETP FOREVER 120 SWAIT EVERFOR
12|DECIMAL
13|-->
14|
15|

```

```

+-----Block      158-----
0|( ANIMATION SEQUENCES TO START SPIRALING ATTACKERS )
1|HEX
2|DATA TFS1 ASM FE5F 0319 SETDC 0006 0604 SETDDC TFF AJMP
3|DATA TFS2 ASM FE6C 0339 SETDC 0806 0604 SETDDC TFF AJMP
4|DATA TFS3 ASM 000E FCFE SETDC 0806 0604 SETDDC TFF AJMP
5|DATA TFS4 ASM 0009 FCBF SETDC 0006 0604 SETDDC TFF AJMP
6|DATA XWS1 ASM FF1F 046F SETDC 0006 0602 SETDDC TFF AJMP
7|DATA XWS2 ASM FEBA 067D SETDC 0007 0603 SETDDC TFF AJMP
8|DATA XWS3 ASM FED5 FC05 SETDC 0006 0601 SETDDC TFF AJMP
9|DATA XWS4 ASM FF8A 06A1 SETDC 0807 0603 SETDDC TFF AJMP
10|TABLE SPTBL TFS1 , TFS2 , TFS3 , TFS4 , XWS1 , XWS2 ,
11|XWS3 , XWS4 , XWS4 , XWS4 , DECIMAL
12|-->
13|
14|
15|
+-----Block      159-----
0|( RETURN SPIRAL STATUS BASED ON SKILL FACTOR )
1|( ON SECOND PASS AND BEYOND SPIRAL IN BOTH DIRECTION )
2|( STATUS BIT IS H.O. OF INTERCEPT MASK BYTE )
3|HEX
4|: PICKDIR SKILLFACTOR @ IF 2 RND 0= IF 80B2 ELSE 0B2 THEN
5|ELSE 0B2 THEN ;
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      160-----
0|( COMMAND TO START SPIRALING ATTACKERS )
1|HEX
2|CODE ACKSUB Y PUSHX, AV LIYD, 0 H LXI,
3|POS Y A LDX, 2 CPI, CY, IF,
4|A ANA, 0=, IF, POS Y INRX, LDAR, 1F ANI, 10 ADI,
5|ATTACKTIMER STA, ELSE, ATTACKTIMER LDED, D A MOV, E ORA,
6|0=, IF, H INX, THEN, THEN,
7|THEN, Y POPX, H PUSH, NEXT
8|DECIMAL
9|: ATTACKCHECK ACKSUB IF ATTACKERSLEFT @ IF ATTACKERSLEFT 1-1
10|ATTACKERSLEFT @ ATXPOS @ ATTACKERSLEFT @ ATYPOS @ 2 POINT
11|SP 17 BOMBTIMER +
12|8 RND SPTBL @ 67 PICKDIR AV @ XNSTART SMS ELSE 0 INVADERSLEFT !
13|THEN THEN ; DECIMAL -->
14|
15|

```

```

+-----Block      161-----
0|( ANIMATION FOR PHOTON TORPEDO ATTACK )
1|DATA PTA ASM
2|FBHCHECK SETI XIWRITE SETR
3|PT1 SETP 15 SWAIT
4|PT2 SETP 15 SWAIT
5|PT3 SETP 20 SWAIT
6|PT4 SETP 40 SWAIT
7|PT5 SETP 10 SWAIT
8|PT6 SETP 10 SWAIT
9|NULPAT SETP 1 SWAIT
10|AHALT
11|DECIMAL -->
12|
13|
14|
15|
+-----Block      162-----
0|( CHECK PHOTON TORPEDOS )
1|F= AVCD CODE AVCK <ASSEMBLE Y PUSHX, 0 H LXI,
2|BOMBTIMER LDED, D A MOV, E ORA, AVCD JRNZ, AV LIYD,
3|PQSRH PQS Y BITX, AVCD JRZ, PQSDE PQS Y BITX, AVCD JRNZ, H INX,
4|DI,
5|LABEL AVCD Y POPX, H PUSH, NEXT ASSEMBLE>
6|( VXL 13 VYL 19 )
7|: PTCHECK AVCK IF
8|SKILLFACTOR @ IF 60 RND 20 ELSE 120 RND 70 THEN + BOMBTIMER ! 0
9|AV @ 13 + @ AV @ 19 + @
10|13 FBVECTOR*0 15 FBVECTOR @
11|PTA 64 162 VMOVE FBL
12|THEN ;
13|DECIMAL -->
14|
15|
+-----Block      163-----
0|( COLOR TABLE -- WAIT FOR ATTACK TO END )
1|HEX
2|DATA M4FBA ASM NULRET SETI 1C05 B005 SETDDC PLAYERANIM AJMP
3|DATA M4COLORS 7 B, 7D B, 0B B, 5F B, 7 B, 7D B, 0B B, 5F B,
4|
5|( WAIT FOR ATTACK END ROUTINE )
6|
7|: WAITOUTAV BEGIN BMS AV @ B@ 80 AND 0= END SHUTUP ;
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      164-----
0|( LINE EFFECT COROUTINE )
1|HEX
2|: LETHREAD GENLINE LSTART 3200 6400 SETLXY 10 10 50 50 SETSF
3|3200 SVCX ! 6400 SVCY !
4|8 LWAIT NULRET LINIT ! A0 LWAIT
5|GENLINE LINIT !
6|30 ATTACKTIMER ! 48 BOMBTIMER !
7|10 10 FF FF SETSF BEGIN -1 LWAIT 0 END ;
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|
+-----Block      165-----
0|( INITIALIZE MISSION 4 - DOGFIGHT IN THE SPACE WARP )
1|HEX : INITM4 0 FLOOD INITMISSIONRAM 34 MISSION !
2|DRAWMISSIONSCREEN 100 5000 408 A" SPACE WARP" COUNT SPOST
3|SKILLFACTOR @ IF 14 2 ELSE 0C 1 THEN SPIRALRATE !
4|ATTACKERSLEFT ! 0 PINTERFLAG ! M4PINTER PHASINTR !
5|100 INVADERSLEFT !
6|' WAITOUTAV REINIT ! SHOWATTACKERS
7|M4FBA FBANIM ! ACTFB
8|GETNODE DUP PV1 ! 0 SWAP !
9|GETNODE DUP AV ! 0 SWAP !
10|-1 ATTACKTIMER ! -1 BOMBTIMER ! 0 LETIMER !
11|' LETHREAD SETLEPC ;
12|DECIMAL -->
13|
14|
15|
+-----Block      166-----
0|( SCAN LOOP AND STARTUP )
1|: M4SCAN LETCK UPDATEALL FIRECHECK ATTACKCHECK PTCHECK
2|PLAYERHITCHECK
3|PHASORINTERCEPTCHECK BMS ;
4|: M4WAIT WTIMER ! BEGIN LETCK UPDATEALL PHASORINTERCEPTCHECK
5|BMS WTIMER @ 0= END ;
6|: M4 INITM4 EMUSIC E2MUSIC ST 5 M4COLORS FUC
7|BEGIN M4SCAN ENDOFFRAME @ END
8|GAMEOVER @ 0= IF 60 M4WAIT THEN 5 FDB ;
9|HEX
10|A5 GSAB U! : M4 GSAB 1+ U!
11|: BEGINGAME STARTGAME SKILLFACTOR ! GSAB 1+ @ DOIT ;
12|DECIMAL ;S
13|
14|
15|

```

```

100|( GAME START ADDRESS BLOCK )
101|( MOM ) DECIMAL DATA MUTHA 6 B, 64 B, QUAD 0 , 0 B, 1110 B, 0 ,
102|( MORE MOM ) 0000 B, 0000 B, 0111 B, 2221 B, 1100 B, 0 B,
103|( MORE MOM ) 0000 B, 0022 B, 1123 B, 3333 B, 2110 B, 3000 B,
104|( LAST MOM ) 2320 B, 0000 B, 0111 B, 1111 B, 1100 B, 0 B,
105|( PHASOR BURST HIT PATTERN )
106|( FIREBALL PATTERNS 1 AND 2 )
107|( FIREBALL PATTERN 3 )
108|( FIREBALL PATTERN 4 )
109|( FIREBALL PATTERN 5 )
110|( ANIMATION SEQUENCE FOR FIREBALL )
111|( BACKGROUND SHIP FLYING - BSF ) HEX
112|( SHIP EXPLOSION - SE ) HEX
113|( FIRE BALL SCORE -FBS, SHIP SHOTOFF - SO ) HEX
114|( MISSIONS- BLACK HOLE EMERGENCE ) HEX
150|( MISSION 5 - ATTACK MOTHER SHIP )
151|( MOTHERSHIP EXPLOSION ANIMATION )
152|( SUBROUTINE TO WRITE MOTHERSHIP PATTERN )
153|( MOTHER SHIP ANIMATION )
154|( CHECK FOR FORCE FIELD INTERCEPT ) HEX
155|( FORCE FIELD INTERCEPT CHECKER CONTINUED )
156|( HAVE PHASOR BURST END WITH A BANG ANIMATION )
157|( MS PHASOR INTERCEPT CHECK ROUTINE )
158|( SUBROUTINE TO SHOOT A FIREBALL )
159|( KAMIKAZE ATTACK STARTER )
160|( CHECK TIMERS AND SHOOT FIREBALL IF APPROPRIATE )
161|( CHUNK SUBROUTINES AND TABLES )
162|( BLOWOFF - START A CHUNK FLYING OFF - DO PATTERN CHANGES )
163|( BLOWOFF - CALCULATE X COORDINATE AND DELTA )
164|( Y COORDINATE PROCESSING )
165|( INITIALIZE ALL THE PARMS FOR INTERRUPT )
166|( CHECK FOR PHASOR - MOTHER INTERCEPT AND BLOWOUT )
167|( PHASOR - MOM - CHECK FOR REACTOR HIT )
168|( PHASOR - MOM CONTINUED )
169|( TRY TO SEND OFF A FRAGMENT ROUTINE )
170|( EXPLODE THE MUTHA SHIP )
171|( CHECK FOR PHASOR HIT MOTHER SHIP )
172|( ANIMATION LIST FOR FIREBASE + COLOR TABLE )
173|( INITIALIZE MISSION 5 - DESTROY THE MOTHER SHIP )
174|( SCAN LOOP AND STARTUP )
181|( GAME START ADDRESS BLOCK )
182|( MOM ) DECIMAL DATA MUTHA 6 B, 64 B, QUAD 0 , 0 B, 2220 B, 0 ,
193|( MORE MOM ) 0000 B, 0000 B, 0222 B, XXXX B, 2220 B, 0 B,
194|( MORE MOM ) 0000 B, 00XX B, 22XX B, 0000 B, 2220 B, 0000 B,
195|( LAST MOM ) XXXX B, 0000 B, 0222 B, 2222 B, 0000 B, 0 B,
196|( PHASOR BURST HIT PATTERN )
197|( FIREBALL PATTERN 3 )
198|( FIREBALL PATTERN 4 )
199|( FIREBALL PATTERN 5 )

```

```

+-----Block 100-----
0|( GAME START ADDRESS BLOCK )
1|DATA GSAB 0 B, 0 ,
2|-->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block 101-----
0|( MOM ) DECIMAL DATA MUTHA 6 B, 64 B, QUAD 0 , 0 B, 1110 B, 0 ,
1|0000 B, 0000 B, 0000 B, 0100 B, 0 B, 0 B,
2|0000 B, 0000 B, 0000 B, 0100 B, 0 B, 0 B,
3|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
4|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
5|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
6|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
7|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
8|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
9|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
10|0000 B, 0000 B, 0000 B, 1110 B, 0 B, 0 B,
11|0000 B, 0000 B, 0001 B, 1111 B, 0 B, 0 B,
12|0000 B, 0000 B, 0001 B, 1111 B, 0 B, 0 B,
13|0000 B, 0000 B, 0011 B, 1211 B, 1000 B, 0 B,
14|0000 B, 0000 B, 0011 B, 1211 B, 1000 B, 0 B,
15|0000 B, 0000 B, 0011 B, 2221 B, 1000 B, 0 B,

```

```

+-----Block 102-----
0|( MORE MOM ) 0000 B, 0000 B, 0111 B, 2221 B, 1100 B, 0 B,
1|0030 B, 0030 B, 0111 B, 2221 B, 1100 B, 0 B,
2|0003 B, 0300 B, 0112 B, 2222 B, 1100 B, 0 B,
3|0000 B, 3000 B, 0112 B, 2222 B, 1100 B, 0 B,
4|0000 B, 0300 B, 1112 B, 2222 B, 1110 B, 0 B,
5|0000 B, 0033 B, 1112 B, 2222 B, 1110 B, 0 B,
6|0000 B, 0000 B, 1122 B, 2222 B, 2110 B, 0 B,
7|0000 B, 0000 B, 1122 B, 2222 B, 2110 B, 0 B,
8|0000 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
9|0000 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
10|0000 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
11|0000 B, 0002 B, 1122 B, 2222 B, 2110 B, 0 B,
12|0000 B, 0002 B, 1122 B, 3332 B, 2110 B, 0 B,
13|0000 B, 0002 B, 1122 B, 3332 B, 2110 B, 0 B,
14|0000 B, 0000 B, 0032 B, 3332 B, 2110 B, 3000 B,
15|0000 B, 0002 B, 1123 B, 3333 B, 2110 B, 3000 B,

```

```

+-----Block      103-----
0|( MORE MOM ) 0000 B, 0022 B, 1123 B, 3333 B, 2110 B, 3000 B,
1|0000 B, 0222 B, 1123 B, 3333 B, 2110 B, 0 B,
2|2220 B, 2222 B, 1123 B, 3333 B, 2110 B, 0 B,
3|2222 B, 2222 B, 1123 B, 3333 B, 2110 B, 0 B,
4|2322 B, 2222 B, 1123 B, 3333 B, 2110 B, 3000 B,
5|2322 B, 2222 B, 1123 B, 3333 B, 2110 B, 3000 B,
6|2322 B, 2220 B, 1123 B, 3333 B, 2110 B, 3000 B,
7|2322 B, 2200 B, 1122 B, 3332 B, 2110 B, 0 B,
8|2322 B, 2000 B, 1122 B, 2322 B, 2110 B, 0 B,
9|2322 B, 0000 B, 1122 B, 2222 B, 2110 B, 0 B,
10|2320 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
11|2320 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
12|2320 B, 0000 B, 1122 B, 2222 B, 2110 B, 3000 B,
13|2320 B, 0000 B, 1112 B, 2222 B, 2110 B, 0 B,
14|2320 B, 0000 B, 1111 B, 1111 B, 1110 B, 0 B,
15|2320 B, 0000 B, 1111 B, 1111 B, 1110 B, 0 B,

```

-->

```

+-----Block      104-----
0|( LAST MOM ) 2320 B, 0000 B, 0111 B, 1111 B, 1100 B, 0 B,
1|2320 B, 0000 B, 0100 B, 0000 B, 0100 B, 0 B,
2|2320 B, 0000 B, 0102 B, 0202 B, 0100 B, 0 B,
3|2320 B, 0000 B, 0102 B, 0202 B, 0100 B, 0 B,
4|2320 B, 0000 B, 0102 B, 0202 B, 0100 B, 0 B,
5|2320 B, 0000 B, 0102 B, 0202 B, 0100 B, 0 B,
6|2320 B, 0000 B, 0002 B, 0202 B, 0 B, 0 B,
7|2320 B, 0000 B, 0002 B, 0202 B, 0 B, 0 B,
8|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
9|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
10|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
11|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
12|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
13|2320 B, 0 B, 0 B, 0 B, 0 B, 0 B,
14|2220 B, 0 B, 0 B, 0 B, 0 B, 0 B,
15|2220 B, 0 B, 0 B, 0 B, 0 B, 0 B, DECIMAL -->

```

```

+-----Block      105-----
0|( PHASOR BURST HIT PATTERN )
1|DATA PBUREXP 2 B, 5 B, QUAD
2|~ 3030 0000 ^
3|~ 0003 0000 ^
4|~ 1113 3000 ^
5|~ 0033 0000 ^
6|~ 0300 0000 ^
7|DECIMAL -->

```

```

8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      106-----
0|( FIREBALL PATTERNS 1 AND 2 )
1|DECIMAL DATA FBL1 4 B, 3 B, QUAD
2|0110 B, 0000 B, 0000 B, 0000 B,
3|3111 B, 1111 B, 1111 B, 0000 B,
4|0130 B, 0000 B, 0000 B, 0000 B,
5|
6|DECIMAL DATA FBL2 4 B, 5 B, QUAD
7|0013 B, 0000 B, 0000 B, 0000 B,
8|0311 B, 1000 B, 0000 B, 0000 B,
9|3111 B, 1111 B, 1111 B, 0000 B,
10|0331 B, 1000 B, 0000 B, 0000 B,
11|0031 B, 0000 B, 0000 B, 0000 B,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      107-----
0|( FIREBALL PATTERN 3 )
1|DECIMAL DATA FBL3 4 B, 7 B, QUAD
2|0003 B, 3100 B, 0000 B, 0000 B,
3|0033 B, 1130 B, 0000 B, 0000 B,
4|0331 B, 1311 B, 0000 B, 0000 B,
5|3333 B, 1111 B, 1111 B, 0000 B,
6|0113 B, 3113 B, 0000 B, 0000 B,
7|0031 B, 1130 B, 0000 B, 0000 B,
8|0001 B, 1100 B, 0000 B, 0000 B,
9|DECIMAL -->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      108-----
0|( FIREBALL PATTERN 4 )
1|DECIMAL DATA FBL4 4 B, 9 B, QUAD
2|0001 B, 1300 B, 0000 B, 0000 B,
3|0011 B, 3330 B, 0000 B, 0000 B,
4|0311 B, 1311 B, 0000 B, 0000 B,
5|1331 B, 1111 B, 1000 B, 0000 B,
6|3133 B, 1131 B, 1111 B, 0000 B,
7|3111 B, 1133 B, 3000 B, 0000 B,
8|0311 B, 1331 B, 0000 B, 0000 B,
9|0013 B, 1310 B, 0000 B, 0000 B,
10|0003 B, 3100 B, 0000 B, 0000 B,
11|DECIMAL -->
12|
13|
14|
15|

```



```

+-----Block      109-----
0|( FIREBALL PATTERN 5 )
1|DECIMAL DATA FBL5 4 B, 11 B, QUAD
2|0000 B, 3110 B, 0000 B, 0000 B,
3|0033 B, 1113 B, 3000 B, 0000 B,
4|0333 B, 3311 B, 3100 B, 0000 B,
5|0331 B, 3331 B, 1100 B, 0000 B,
6|3311 B, 1311 B, 1330 B, 0000 B,
7|1113 B, 1111 B, 3331 B, 0000 B,
8|1111 B, 1131 B, 3313 B, 0000 B,
9|0131 B, 3133 B, 3300 B, 0000 B,
10|0333 B, 1113 B, 3300 B, 0000 B,
11|0011 B, 1133 B, 1000 B, 0000 B,
12|0000 B, 1130 B, 0000 B, 0000 B,
13|DECIMAL -->
14|
15|

+-----Block      110-----
0|( ANIMATION SEQUENCE FOR FIREBALL )
1|HEX SUBR SETFBD XCHG, LDAR, 7F ANI, A C MOV, 0 B MVI,
2|40 H LXI, A ANA, B DSBC, H DAD, H DAD,
3|L VDYL X STX, H VDYH X STX, XCHG, RET,
4|DATA AFIREBALL ASM XADDWRITE SETR NULPAT SETFP -FB 0 SETDC
5|SETFBD ASMCALL
6|3 0 SETDDC PBURST SETP 5 SWAIT 0 -1 DISPL FBL1 SETP 6 SWAIT
7|0 -1 DISPL
8|2 AREPEAT FBL2 SETP 20 SETM 3 SWAIT A0 SETM 3 SWAIT ALOOP
9|0 -1 DISPL
10|3 AREPEAT FBL3 SETP 20 SETM 3 SWAIT A0 SETM 3 SWAIT ALOOP
11|0 -1 DISPL 0 0 SETDDC
12|3 AREPEAT FBL4 SETP 20 SETM 3 SWAIT A0 SETM 3 SWAIT ALOOP
13|0 -1 DISPL
14|FOREVER FBL5 SETP 20 SETM 3 SWAIT A0 SETM 3 SWAIT EVERFOR
15|DECIMAL -->

+-----Block      111-----
0|( BACKGROUND SHIP FLYING - BSF ) HEX
1| DATA BSFSCORE ASM
2| 23 MASTER 1 2 52 10 RAMBLE 1 -2 3F MOVESOUND
3| 99 ABVOLS 09 MCVOLS 4 FF CTONE HERE
4| 4 FF FE FD NOTES 4 FE FC FA NOTES 4 FD FA F7 NOTES
5| 4 FC F8 F4 NOTES 4 F2 F6 F1 NOTES 4 FA F4 EE NOTES
6| 4 F9 F2 EB NOTES LDPCC
7|: BSF BSFSCORE DEMUSIC ;
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      112-----
0| ( SHIP EXPLOSION - SE ) HEX
1| DATA SE1SCORE ASM
2| #C3 #D3 #E3 TONES HERE 2 1 0 MOVESOUND 66 ABVOLS 6 MCVOLS
3| 80 MASTER 1 -1 80 2 RAMBLE 1 COUNTLIMITS PLAY
4| 10 NOISE 6 1 F 2 RAMBLE 1 1 3F MOVESOUND 5 COUNTLIMITS
5| FF ABVOLS 3F MCVOLS PLAY FF 1F 16 -1 0 0F MOVEVOLS
6| 2 1 B0 F RAMBLE B7 0 0 0 MOVENOISE 1 COUNTLIMITS PLAY QUIET
7| DATA SE2SCORE ASM 6 DURATION
8| #F3 #G3 #A3 TONES LDPCC
9| : SE SE2SCORE P2MUSIC SE1SCORE PMUSIC ;
10| -->
11|
12|
13|
14|
15|
+-----Block      113-----
0| ( FIRE BALL SCORE -FBS, SHIP SHOTTOFF - SO ) HEX
1| DATA FBSCORE ASM
2| 10 MASTER 2 1 30 10 RAMBLE 1 COUNTLIMITS #DS3 #E3 #D3 TONES
3| 2 -1 3F MOVESOUND 22 12 10 1 0 0C MOVEVOLS PLAY
4| 0 2 3 6C MOVENOISE 2 3 8F 30 RAMBLE 1 COUNTLIMITS PLAY
5| BSFSCORE LDPCC ( jump to background )
6| : FBS E2MUSIC FBSCORE B2MUSIC ;
7| DATA SOSCORE ASM
8| 10 MASTER 2 -2 10 6 RAMBLE 5 0 0 0 MOVENOISE 1 -3 3F MOVESOUND
9| BB ABVOLS 1B MCVOLS 4 FF FE 20 NOTES
10| 4 FE FC 1C NOTES 4 FD FA 1A NOTES QUIET
11| : SO EMUSIC SOSCORE BMUSIC ; -->
12|
13|
14|
15|
+-----Block      114-----
0| ( MISSIONS- BLACK HOLE EMERGENCE ) HEX
1| DATA BH1SCORE ASM
2| 11 40 62 TONES 10 MASTER 1 4 C0 10 RAMBLE 1 COUNTLIMITS
3| 2 1 0 MOVESOUND 0 1 4 B0 MOVENOISE 88 ABVOLS 1C MCVOLS PLAY
4| C0 MASTER 1 -8 C0 2 RAMP B0 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
5| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
6| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
7| 2 MASTER 3 1 FF 2 RAMBLE 1 COUNTLIMITS PLAY QUIET
8| DATA BH2SCORE ASM
9| 13 30 50 TONES 10 MASTER 1 4 C0 10 RAMBLE 1 COUNTLIMITS
10| 2 1 0 MOVESOUND 0 1 4 B0 MOVENOISE 88 ABVOLS 1C MCVOLS PLAY
11| C0 MASTER 1 -8 C0 2 RAMP 30 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
12| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
13| 80 MASTER 1 -8 80 2 RAMP 70 1 -8 0 MOVENOISE 1 COUNTLIMITS PLAY
14| 2 MASTER 3 1 FF 2 RAMBLE 1 COUNTLIMITS PLAY QUIET
15| : BH BH1SCORE PMUSIC BH2SCORE P2MUSIC ; DECIMAL 10

```

```

+-----Block      150-----
0|( MISSION 5 - ATTACK MOTHER SHIP )
1|DECIMAL
2|0 V= MSHITF 0 V= MSIYC 0 V= MSV 0 V= CHUNKXTBL
3|0 V= HITXC 0 V= HITYC 0 V= CHUNKTIME
4|0 V= VGF1 0 V= VGF2
5|386 BA= MOMPAT
6|: COPYMOM 386 0 DO I MUTHA + B@ I MOMPAT B! LOOP ;
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      151-----
0|( MOTHERSHIP EXPLOSION ANIMATION )
1|DATA MSDIE ASM 0 0 SETDC 0 MOMPAT SETP 1 SWAIT
2|8 26 DISPL FBEXP1 SETP 20 SWAIT
3|-3 -4 DISPL FBEXP2 SETP 20 SWAIT
4|-1 -2 DISPL FBEXP3 SETP 20 SWAIT
5|FBEXP4 SETP 20 SWAIT NULPAT SETP 1 SWAIT AHALT
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      152-----
0|( SUBROUTINE TO WRITE MOTHERSHIP PATTERN )
1|SUBR WMOM PQSFRZ PQS X BITX, 0=, IF, TBCALC CALL, B PUSH,
2|PQSD E PQS X BITX, 0=, IF, verase CALL, ELSE,
3|PQSD E PQS X RESX,
4|THEN, ( ZAPAT CALL, ) ( ZAP HOLES IN PATTERN, IF NEEDED )
5| B POP, VECTDD CALL, sup CALL, PQSDW PQS X BITX, 0=,
6|IF, vwrite CALL, ELSE, PQSDW PQS X RESX, PQSD E PQS X SETX,
7|THEN, ELSE, 0 PQTS X MVIX, THEN, KILLOFF JMP,
8|-->
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      153-----
0|( MOTHER SHIP ANIMATION )
1|HEX
2|DATA AMUTHA ASM WMOM SETR 0 MOMPAT SETP
3|50 SETXZW 4000 SETXC 1000 SETYC
4|FOREVER
5|6 AREPEAT A0 SETM -2 40 SETDC 0 2 SETDDC 60 SWAIT 0 -2 SETDDC
6|60 SWAIT 20 SETM -2 -40 SETDC 0 -2 SETDDC 60 SWAIT 0 2 SETDDC
7|60 SWAIT ALOOP
8|6 AREPEAT A0 SETM 2 40 SETDC 0 2 SETDDC 60 SWAIT 0 -2 SETDDC
9|60 SWAIT 20 SETM 2 -40 SETDC 0 -2 SETDDC 60 SWAIT 0 2 SETDDC
10|60 SWAIT ALOOP EVERFOR
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      154-----
0|( CHECK FOR FORCE FIELD INTERCEPT ) HEX
1|F= FFSL F= FFOK F= FFZL F= NOFF
2|SUBR FFCHECK <ASSEMBLE
3|FFLAG LDA, A ANA, RZ, VYH X C LDX, 0 B MVI,
4|0 FIELDADR H LXI, B DAD, L E MOV, H D MOV, 3 B MVI,
5|LABEL FFSL M A MOV, A ANA, FFOK JRNZ, H INX, FFSL DJNZ,
6|A ANA, RET,
7|LABEL FFOK RRC, RRC, 3F ANI, A B MOV, FFBias LDA, B ADD,
8|VXH X SUBX,
9|4 ADI, 7 CPI, NOFF JNC, C DCR, D DCX, D PUSH,
10|C L MOV, 0 H MVI, H DAD, H DAD, H DAD, H DAD,
11|L C MOV, H B MOV, H DAD, H DAD, B DAD, FFBias LBCD, B DAD,
12|-->
13|
14|
15|

```

```

+-----Block      155-----
0|( FORCE FIELD INTERCEPT CHECKER CONTINUED )
1|5 B MVI,
2|LABEL FFZL XTHL, M A MOV, A ANA, 0<>, IF,
3|A C MOV, 3 ANI, 20 ORI, MAGIC OUT, A XRA, A M MOV,
4|H INX, XTHL, C A MOV, RRC, RRC,
5|3F ANI, A E MOV, 0 D MVI, XCHG, D DAD, 0FF M MVI,
6|H INX, 0 M MVI, XCHG, ELSE, H INX, XTHL, THEN,
7|50 D LXI, D DAD, FFZL DJNZ,
8|H POP, 1 A MVI, A ANA, RET,
9|LABEL NOFF A XRA, SET,
10|ASSEMBLE>
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      156-----
0|( HAVE PHASOR BURST END WITH A BANG ANIMATION )
1|DATA APBEXP ASM 0 0 SETDC 8 -2 DISPL
2|NULPAT SETFP FBUREXP SETP 3 SWAIT AHALT
3|DECIMAL -->
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      157-----
0|( M5 PHASOR INTERCEPT CHECK ROUTINE )
1|HEX F= CFF F= PIS
2|SUBR MSPINTER <ASSEMBLE
3|MSV LIYD, CHECKVEC CALL, CFF JRZ,
4|PQSFRZ PQS Y SETX,
5|VXL Y L LDX, VXH Y H LDX, HITXC SHLD,
6|VYL Y L LDX, VYH Y H LDX, HITYC SHLD,
7|VYH X A LDX, MSIYC STA, PIS JMPR,
8|LABEL CFF FFCHECK CALL, PIS JRNZ,
9|9 C MVI, CHECKALL CALL, RZ, PQSRH PQS Y RESX, PQSDW PQS Y SETX,
10|VYL Y L LDX, VYH Y H LDX, PINTERX SHLD, VXL Y L LDX,
11|VXH Y H LDX, PINTERX SHLD, 1 A MVI, PINTERFLAG STA,
12|VRACK Y A LDX, PINTERN STA,
13|LABEL PIS APBEXP H LXI, CRASHA CALL, PQSFRZ PQS X SETX,
14|XAWRITE H LXI, L PQRH X STX, H PQRH X STX, RET,
15|ASSEMBLE> DECIMAL -->

+-----Block      158-----
0|( SUBROUTINE TO SHOOT A FIREBALL )
1|HEX
2|SUBR SHOOTFB MSV LIYD, VXL Y C LDX, VXH Y B LDX,
3|B DCR, B PUSH, VYL Y L LDX, VYH Y H LDX,
4|2000 D LXI, D DAD, H PUSH, AFIREBALL H LXI,
5|H PUSH, B A MOV, RLC, 0A SUI, A L MOV, 0 H MVI,
6|H PUSH,
7|04A4 H LXI, H PUSH,
8|SKILLFACTOR LDA, A ANA, 0=, 1F, LDAR, 3F ANI, 40 ADI, ELSE,
9|LDAR, 1F ANI, 10 ADI, THEN, BOMBTIMER STA,
10|VXH Y A LDX, 8 SUI, FBTIMER STA,
11|XYVSTART JMP,
12|
13|CODE FBshoot X PUSHX, Y PUSHX, EXX, SHOOTFB CALL,
14|EXX, Y POPX, X POPX, NEXT
15|DECIMAL -->

```

```

+-----Block      159-----
0|( KAMIKAZE ATTACK STARTER )
1|HEX SUBR KAMISTART
2|LDAR, 1 ANI, 0=, IF, VGF1 LIXD, ELSE, VGF2 LIXD, THEN,
3|DI, PQSRH PQS X BITX, RZ, ASFLOK VAUXS X BITX, RZ,
4|VYH X A LDX, 20 SUI, 90 CPI, RNC, LDAR, 1 ANI,
5|0=, IF, KAMIATL H LXI, ELSE, KAMIATR H LXI, THEN,
6|ASFLOK VAUXS X RESX, CRASHA CALL, LDAR, 1F ANI, 18 ADI,
7|BOMBTIMER STA, PLAYKBS JMP,
8|
9|CODE LAUNCHKAMI X PUSHX, Y PUSHX, B PUSH, KAMISTART CALL, EI,
10|B POP, Y POPX, X POPX, NEXT
11|
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      160-----
0|( CHECK TIMERS AND SHOOT FIREBALL IF APPROPRIATE )
1|: LAUNCHFIREBALL FBS FBShoot ;
2|: CHECKFIREBALL
3|BOMBTIMER @ 0 = IF SKILLFACTOR @ IF 2 RND IF LAUNCHKAMI
4|ELSE LAUNCHFIREBALL THEN ELSE LAUNCHFIREBALL THEN THEN ;
5|DECIMAL -->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      161-----
0|( CHUK SUBROUTINES AND TABLES )
1|HEX DATA DXLUT -80 , -40 , -10 , 10 , 40 , 80 ,
2|DATA DXALLDOWN -80 , -80 , -80 , -80 , -80 , -80 ,
3|DATA DXHISKILL -C0 , -C0 , -C0 , -C0 , -C0 , -C0 ,
4|DATA NULANIM ASM AHALT
5|SUBR MOMR2A ( B = PATTERN Y C = PATTERN X IY = MOM ADDR )
6|MRFLIP VMAGIC Y BITX, 0<>, IF, 3F A MVI, B SUB, A L MOV,
7|ELSE, B L MOV, THEN,
8|0 H MVI, H DAD,
9|L E MOV, H D MOV, H DAD, D DAD,
10|C E MOV, 0 D MVI, D DAD,
11|2 MOMPAT D LXI, D DAD, RET,
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      162-----
0|( BLOWOFF - START A CHUNK FLYING OFF - DO PATTERN CHANGES )
1|( B = PATTERN X C = PATTERN Y ) HEX
2|SUBR BLOWOFF DI, MSV LIYD,
3|B PUSH, getnode CALL,
4|H PUSH, X POPX, CLRVEC CALL,
5|B POP, MOMR2A CALL, B PUSH, H PUSH,
6|X PUSHX, H POP, VASTKS D LXI, D DAD,
7|L VPATL X STX, H VPATH X STX,
8|2 M MVI, H INX, 3 M MVI, H INX,
9|XCHG, H POP, 3 B MVI,
10|BEGIN, M A MOV, D STAX, 0 M MVI, D INX, D INX,
11|MRFLIP VMAGIC Y BITX, 0=, IF,
12|H INX, H INX, H INX, H INX, H INX, H INX,
13|ELSE, H DCX, H DCX, H DCX, H DCX, H DCX, H DCX,
14|THEN, LOOP,
15|B POP, -->

```

```

+-----Block      163-----
0|( BLOWOFF - CALCULATE X COORDINATE AND DELTA )
1|HITXC LHLD,
2|C D MOV, 0 E MVI, D DAD,
3|L VXL X STX, H VXH X STX,
4|
5|C A MOV, RLC, A E MOV, 0 D MVI,
6|CHUNKXTBL LHLD, D DAD,
7|M E MOV, H INX, M D MOV,
8|VDXL Y L LDX, VDXH Y H LDX, D DAD,
9|L VDXL X STX, H VDXH X STX,
10|-->
11|
12|
13|
14|
15|

```

```

+-----Block      164-----
0|( Y COORDINATE PROCESSING )
1|HITYC LHLD,
2|B D MOV, 0 E MVI, D DAD,
3|L VYL X STX, H VYH X STX,
4|
5|B L MOV, 0 H MVI, -20 D LXI,
6|D DAD, H DAD, H DAD, H DAD, ( H DAD, )
7|VDYL Y E LDX, VDYH Y D LDX, D DAD,
8|L VDYL X STX, H VDYH X STX,
9|-->
10|
11|
12|
13|
14|
15|

```

```

+-----Block      165-----
0|( INITIALIZE ALL THE PARMS FOR INTERRUPT )
1|20 VMAGIC X MVIX,
2|8 VIDENT X MVIX, 46 VRACK X MVIX,
3|CHUNKTIME LHLD, L VTLL X STX, H VTLL X STX,
4|78 VATMR X MVIX,
5|0A4 PQS X MVIX,
6|30 VXZW X MVIX,
7|NULPAT H LXI, L VFNLPL X STX, H VFNLPH X STX,
8|XAWRITE H LXI, L PQRL X STX, H PQRH X STX,
9|NULANIM H LXI, L VPCL X STX, H VPCH X STX,
10|vwrite CALL, STARTVEC CALL, RET,
11|DECIMAL -->
12|
13|
14|
15|

+-----Block      166-----
0|( CHECK FOR PHASOR - MOTHER INTERCEPT AND BLOWOUT )
1|F= MHSF F= MHSF F= MHNG F= MHGB F= MHNG1 F= RSL F= REH F= NHIT
2|HEX CODE MPHC <ASSEMBLE
3|EXX, X PUSHX, Y PUSHX, MSV LIYD,
4|0 H LXI, MSIYC LDA, A ANA, MHNG1 JZ,
5|VYH Y SUBX, 40 CPI, MHNG JNC,
6|A B MOV, 0 C MVI,
7|LABEL MHSF MOMR2A CALL, M A MOV, A ANA, MHSF JRNZ,
8|C INR, C A MOV, 6 CPI, MHSF JRC, MHNG JMP,
9|-->
10|
11|
12|
13|
14|
15|

+-----Block      167-----
0|( PHASOR - MOM - CHECK FOR REACTOR HIT )
1|LABEL MHSF
2|MRFLIP VMAGIC Y BITX, 0<>, IF, 3F A MVI, B SUB, ELSE, B A MOV,
3|THEN, 1C CPI, NHIT JC, 2A CPI, NHIT JNC,
4|C A MOV, 2 CPI, NHIT JC, 4 CPI, NHIT JNC, 4 E MVI, M D MOV,
5|LABEL RSL D A MOV, RLC, RLC, A D MOV, 3 ANI, 3 CPI, REH JRZ,
6|E DCR, RSL JRNZ, NHIT JMP,
7|LABEL REH DI, MSHIF STA, Y PUSHX, XTIX,
8|B PUSH, MSDIE H LXI,
9|CRASHA CALL, XAWRITE H LXI, L PQRL X STX, H PQRH X STX,
10|POSDE PQS X SETX, 0 POP, X POPX,
11|-->
12|
13|
14|
15|

```



```

+-----Block      168-----
0|( PHASOR - MOM CONTINUED )
1|LABEL NHIT B A MOV, A DCR, 0<, IF, A XRA, ELSE,
2|3E CPI, CY~, IF, 3D A MVI, THEN, THEN, A B MOV,
3|BLOWOFF CALL, 1 H LXI, MHGB JMPR,
4|LABEL MHNG 0 H LXI,
5|LABEL MHGB A XRA, MSIYC STA, PQSFRZ PQS Y RESX,
6|LABEL MHNG1 Y POPX, X POPX, H PUSH, EXX, NEXT
7|ASSEMBLE> DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      169-----
0|( TRY TO SEND OFF A FRAGMENT ROUTINE )
1|HEX CODE SENDFRAG EXX, D POP, E B MOV, D POP, E C MOV,
2|X PUSHX, Y PUSHX,
3|MSV LIYD, MOMR2A CALL,
4|M A MOV, A ANA, 0<>, IF,
5|FREELIST LHLD, H A MOV, L ORA, 0<>, IF,
6|BLOWOFF CALL, THEN, THEN,
7|Y POPX, X POPX, EXX, NEXT
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|
+-----Block      170-----
0|( EXPLODE THE MUTHA SHIP )
1|HEX
2|: EXPLODEMUTHA ERASEFF
3|DXLUT CHUNKXTBL ! 20 CHUNKTIME !
4|WRONLY LINIT ! HITXC @ 280 + DUP MUTHAX !
5|HITYC @ 2000 + DUP MUTHAY ! SETLXY
6|6 6 30 30 SETSF DI UPDATEALL EI SE BMS 1 STARZ OUTP
7|10 0 DO 8 0 DO 3 RND 1+ 30 RND 8 + SENDFRAG UPDATEALL BMS
8|WVI 1 0 OUTP 1 4 OUTP EI LOOP 6 6 I 3 * 40 + DUP SETSF LOOP
9|9 STARZ OUTP WVI 7 0 OUTP 7 4 OUTP EI
10|100 0 DO 6 RND CC RND 2 + SENDFRAG UPDATEALL BMS LOOP
11|TO 1 ENDOFFRAME !
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      171-----
0|( CHECK FOR PHASOR HIT MOTHER SHIP )
1|HEX
2|: PSCORECHECK MPHCL IF
3|20 UPDATESCORE
4|MSHITF @ IF 1000 UPDATESCORE EXPLODEMUTHA ELSE SO THEN
5|THEN ; DECIMAL -->
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      172-----
0|( ANIMATION LIST FOR FIREBASE + COLOR TABLE )
1|HEX
2|SUBR MSINTER MSHITF LDA, A ANA, RNZ, 0D C MVI, CHECKALL CALL,
3|RZ, PQSRH PQS Y RESX, PQSDW PQS Y SETX, EXPLODEFB CALL, RET,
4|
5|DATA MSFBA ASM MSINTER SETI 1F05 B005 SETDDC PLAYERANIM
6|AJMP
7|DATA MSCOLORS 7 B, 7D B, 0B B, 5F B, 7 B, 7D B, 0B B, 5F B,
8|DECIMAL -->
9|
10|
11|
12|
13|
14|
15|
+-----Block      173-----
0|( INITIALIZE MISSION 5 - DESTROY THE MOTHER SHIP )
1|HEX : INITM5 0 FLOOD INITMISSIONRAM 35 MISSION !
2|DRAWMISSIONSCREEN 100 5000 408 A" DEATH SHIP" COUNT SPOST
3|0C FFBIAS ! 2 INITFF
4|0 PINTERFLAG ! 0 MSIYC ! 0 MSHITF !
5|COPYMOM MSPINTER PHASINTR ! -1 INVADERSLEFT !
6|MSFBA FBANIM ! ACTFB
7|GETNODE DUP PV1 ! 0 SWAP ! GETNODE MSV !
8|60 BOMBTIMER ! 0 BSE REINIT !
9|SKILLFACTOR 0 IF DYNISKILL ELSE DHALLOWN THEN CHUNXXTEL !
10|70 CHUNKTIME ! DRAWFF ;
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      174-----
0|( SCAN LOOP AND STARTUP )
1|: M5SCAN FIRECHECK ( EXPLODECHECK ) CHECKFIREBALL
2|PLAYERHITCHECK PHASORINTERCEPTCHECK PSCORECHECK BMS ;
3|HEX
4|: M5 INITM5 AMUTHA 0 B2 MSV @ XVSTART SHUTUP BH BSF
5|SKILLFACTOR @ IF GETNODE VGF1 ! GETNODE VGF2 !
6|MSV @ 0600 0 AKGORF 43 1B2 VGF1 @ FSTART
7|MSV @ 0600 3600 AKGORF 43 1B2 VGF2 @ FSTART THEN
8|5 M5COLORS FUC
9|BEGIN M5SCAN
10|ENDOFFRAME @ END 4 FDB ;
11|HEX A5 GSAB UI ' M5 GSAB 1+ UI
12|: BEGINGAME STARTGAME SKILLFACTOR ! GSAB 1+ @ DOIT ;
13|DECIMAL ;S
14|
15|

```

```

+-----Block      191-----
0|( GAME START ADDRESS BLOCK )
1|DATA GSAB 0 B, 0 ,
2|-->
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      192-----
0|( MOM ) DECIMAL DATA MUTHA 6 B, 64 B, QUAD 0 , 0 B, &&&0 B, 0 ,
1|0000 B, 0000 B, 0000 B, 0&00 B, 0 B, 0 B,
2|0000 B, 0000 B, 0000 B, 0&00 B, 0 B, 0 B,
3|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
4|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
5|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
6|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
7|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
8|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
9|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
10|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
11|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
12|0000 B, 0000 B, 0000 B, &&&0 B, 0 B, 0 B,
13|0000 B, 0000 B, 0000 B, &&&0 B, &000 B, 0 B,
14|0000 B, 0000 B, 0000 B, &&&0 B, &000 B, 0 B,
15|0000 B, 0000 B, 0000 B, &&&0 B, &000 B, 0 B,

```

+-----Block 193-----

0 (MORE MOM) 0000 B, 0000 B, 0&&& B, ***& B, &&00 B, 0 B,
 1 00\$0 B, 00\$0 B, 0&&& B, ***& B, &&00 B, 0 B,
 2 000\$ B, 0\$00 B, 0&&& B, ***& B, &&00 B, 0 B,
 3 0000 B, \$000 B, 0&&& B, ***& B, &&00 B, 0 B,
 4 0000 B, 0\$00 B, &&&& B, ***& B, &&00 B, 0 B,
 5 0000 B, 00\$0 B, &&&& B, ***& B, &&00 B, 0 B,
 6 0000 B, 0000 B, &&&& B, ***& B, *&&0 B, 0 B,
 7 0000 B, 0000 B, &&&& B, ***& B, *&&0 B, 0 B,
 8 0000 B, 0000 B, &&&& B, ***& B, *&&0 B, \$000 B,
 9 0000 B, 0000 B, &&&& B, ***& B, *&&0 B, \$000 B,
 10 0000 B, 0000 B, &&&& B, ***& B, *&&0 B, \$000 B,
 11 0000 B, 000* B, &&&& B, *\$*** B, *&&0 B, 0 B,
 12 0000 B, 000* B, &&&& B, \$\$\$* B, *&&0 B, 0 B,
 13 0000 B, 000* B, &&&& B, \$\$\$* B, *&&0 B, 0 B,
 14 0000 B, 0000 B, 00\$\$\$ B, \$\$\$* B, *&&0 B, \$000 B,
 15 0000 B, 000* B, &&&& B, \$\$\$* B, *&&0 B, \$000 B, -->

+-----Block 194-----

0 (MORE MOM) 0000 B, 00** B, &&*\$ B, \$\$\$* B, *&&0 B, \$000 B,
 1 0000 B, 0*** B, &&*\$ B, \$\$\$* B, *&&0 B, 0 B,
 2 ***0 B, ***\$ B, &&*\$ B, \$\$\$* B, *&&0 B, 0 B,
 3 ***\$ B, ***\$ B, &&*\$ B, \$\$\$* B, *&&0 B, 0 B,
 4 ***\$ B, ***\$ B, &&*\$ B, \$\$\$* B, *&&0 B, \$000 B,
 5 ***\$ B, ***\$ B, &&*\$ B, \$\$\$* B, *&&0 B, \$000 B,
 6 ***\$ B, ***0 B, &&*\$ B, \$\$\$* B, *&&0 B, \$000 B,
 7 ***\$ B, **00 B, &&*\$ B, \$\$\$* B, *&&0 B, 0 B,
 8 ***\$ B, *000 B, &&*\$ B, \$\$\$* B, *&&0 B, 0 B,
 9 ***\$ B, 0000 B, &&*\$ B, ***\$ B, *&&0 B, 0 B,
 10 ***0 B, 0000 B, &&*\$ B, ***\$ B, *&&0 B, \$000 B,
 11 ***0 B, 0000 B, &&*\$ B, ***\$ B, *&&0 B, \$000 B,
 12 ***0 B, 0000 B, &&*\$ B, ***\$ B, *&&0 B, \$000 B,
 13 ***0 B, 0000 B, &&&& B, ***\$ B, *&&0 B, 0 B,
 14 ***0 B, 0000 B, &&&& B, &&&& B, &&&0 B, 0 B,
 15 ***0 B, 0000 B, &&&& B, &&&& B, &&&0 B, 0 B, -->

+-----Block 195-----

0 (LAST MOM) *\$*0 B, 0000 B, 0&&& B, &&&& B, &&00 B, 0 B,
 1 *\$*0 B, 0000 B, 0&&0 B, 0000 B, 0&&0 B, 0 B,
 2 *\$*0 B, 0000 B, 0&0* B, 0*0* B, 0&&0 B, 0 B,
 3 *\$*0 B, 0000 B, 0&0* B, 0*0* B, 0&&0 B, 0 B,
 4 *\$*0 B, 0000 B, 0&0* B, 0*0* B, 0&&0 B, 0 B,
 5 *\$*0 B, 0000 B, 0&0* B, 0*0* B, 0&&0 B, 0 B,
 6 *\$*0 B, 0000 B, 000* B, 0*0* B, 0 B, 0 B,
 7 *\$*0 B, 0000 B, 000* B, 0*0* B, 0 B, 0 B,
 8 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 9 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 10 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 11 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 12 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 13 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 14 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B,
 15 *\$*0 B, 0 B, 0 B, 0 B, 0 B, 0 B, DECIMAL -->

```

+-----Block      196-----
0|( PHASOR BURST HIT PATTERN )
1|DATA PBUREXP 2 B, 5 B, QUAD
2|~ 3030 0000 ^
3|~ 0003 0000 ^
4|~ 1113 3000 ^
5|~ 0033 0000 ^
6|~ 0300 0000 ^
7|DECIMAL -->
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      197-----
0|( FIREBALL PATTERN 3 )
1|DECIMAL DATA FBL3 4 B, 7 B, QUAD
2|0003 B, 3100 B, 0000 B, 0000 B,
3|0033 B, 1130 B, 0000 B, 0000 B,
4|0331 B, 1311 B, 0000 B, 0000 B,
5|3333 B, 1111 B, 1111 B, 0000 B,
6|0113 B, 3113 B, 0000 B, 0000 B,
7|0031 B, 1130 B, 0000 B, 0000 B,
8|0001 B, 1100 B, 0000 B, 0000 B,
9|DECIMAL -->
10|
11|
12|
13|
14|
15|
+-----Block      198-----
0|( FIREBALL PATTERN 4 )
1|DECIMAL DATA FBL4 4 B, 9 B, QUAD
2|0001 B, 1300 B, 0000 B, 0000 B,
3|0011 B, 3330 B, 0000 B, 0000 B,
4|0311 B, 1311 B, 0000 B, 0000 B,
5|1331 B, 1111 B, 1000 B, 0000 B,
6|3133 B, 1131 B, 1111 B, 0000 B,
7|3111 B, 1133 B, 0000 B, 0000 B,
8|0311 B, 1331 B, 0000 B, 0000 B,
9|0013 B, 1310 B, 0000 B, 0000 B,
10|0003 B, 3100 B, 0000 B, 0000 B,
11|DECIMAL -->
12|
13|
14|
15|

```

```
+-----Block      199-----  
0|( FIREBALL PATTERN 5 )  
1|DECIMAL DATA FBL5 4 B, 11 B, QUAD  
2|0000 B, 3110 B, 0000 B, 0000 B,  
3|0033 B, 1113 B, 3000 B, 0000 B,  
4|0333 B, 3311 B, 3100 B, 0000 B,  
5|0331 B, 3331 B, 1100 B, 0000 B,  
6|3311 B, 1311 B, 1330 B, 0000 B,  
7|1113 B, 1111 B, 3030 B, 0000 B,  
8|1111 B, 1131 B, 0010 B, 0000 B,  
9|0131 B, 3133 B, 0000 B, 0000 B,  
10|0333 B, 1113 B, 0000 B, 0000 B,  
11|0011 B, 1133 B, 0000 B, 0000 B,  
12|0000 B, 1130 B, 0000 B, 0000 B,  
13|DECIMAL -->  
14|  
15|
```

```

100|( CROSS COMPILE TACK GAME ON SYSTEM END ) DECIMAL
101|( MASTER CONTROL PROGRAM )
102|( DISPLAY PLAYER UP )
103|( OTHER NEAT SUBROUTINES )
104|( OTHER NEAT SUBROUTINES )
105|( DO A ONE PLAYER GAME )
106|( TWO PLAYER GAME )
107|( GAME START UP AND HAVE SOME FUN )
121|( MASTER CONTROL PROGRAM )
122|( DISPLAY PLAYER UP )
123|( OTHER NEAT SUBROUTINES )
124|( TWO PLAYER GAME )
125|( GAME START UP AND HAVE SOME FUN )
150|( CHARACTER VECTORING ROUTINE )
151|( WONDERFULL TEST CHARACTER INSANITY ROUTINE )
160|( FIREBASE EXPLOSION 5 )
161|( FBEXPS CONTINUED )
162|( FIRE BASE EXPLOSION 6 )
163|( FIRE BASE EXPLOSION 6 CONTINUED )
170|( NORMAL BCD ADDITION )
171|( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBOD6 )

```

```

+-----Block      100-----
0|( CROSS COMPILE TACK GAME ON SYSTEM END ) DECIMAL
1|HEX 3900 DF ! DECIMAL
2|101 B: LOAD
3|XCEND XCSTAT
4|xcsys xc DECIMAL ;S
5|DECIMAL 305 300 <<
6|HEX 300 200 100 0 4 0 << DECIMAL J HEXLIST >> >>
7|CR PAGE CR PAGE CR ;S
8|
9|
10|
11|
12|
13|
14|
15|
+-----Block      101-----
0|( MASTER CONTROL PROGRAM )
1|HEX ( MISSION START ADDRESSES TABLE )
2|TABLE MSATBL 8000 , 9000 , 0A000 , 0A800 , 0B000 ,
3|( CRUDE GAME OVER COLORS )
4|DATA MCCOLORS 0 B, 7D B, 0B B, 5A B, 0 B, 7D B, 0B B, 5A B,
5|: GOFAME DI 0 FLOOD INITMISSIONRAM DRAWMISSIONSCREEN
6|0 STARZ OUTP DI ;
7|: GOC MCCOLORS COLOR ;
8|: CLRLITES 27 20 DO 0 I OUTP LOOP ;
9|: LITEMISSION CLRLITES 1 MISSION @ 11 - OUTP SKILLFACTOR @
10|IF 1 25 OUTP THEN ;
11|: DOFRAME 0 FLOOD LITEMISSION MISSION @ 31 - MSATBL @ DUP B@
12|A5 = IF 1+ @ DOIT ELSE DROP THEN 0 26 OUTP ;
13|DECIMAL -->
14|
15|
+-----Block      102-----
0|( DISPLAY PLAYER UP )
1|HEX 428 C= DIPS DECIMAL
2|: DISPU GOFAME DI
3|240 76 XY DIPS A" PLAYER" COUNT SPOST
4|PLAYERUP @ IF 200 86 XY DIPS A" TWO" COUNT SPOST ELSE
5|200 86 XY DIPS A" ONE" COUNT SPOST THEN ;
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```



```

+-----Block      103-----
0|( OTHER NEAT SUBROUTINES )
1|HEX
2|: BUMPMISS MISSIONCTR BCDBUMP MISSION 1+! MISSION @ 36 = IF
3|SKILLFACTOR @ 0= IF FBCOUNTER 1+! OTHERFBCTR 1+! THEN
4|SKILLFACTOR 1+!
5|OTHERSKILLF 1+! 31 MISSION ! THEN ;
6|: BUMPCHECK PLAYERUP @ 0= IF BUMPMISS ELSE P1ACT @ GAMEOVER @
7|OR 0= IF BUMPMISS THEN THEN ;
8|: MYSTATE PLAYERUP @ IF P2ACT ELSE P1ACT THEN ;
9|: OTHERSTATE PLAYERUP @ IF P1ACT ELSE P2ACT THEN ;
10|: SWAPPLAYER OTHERSTATE @ IF
11|PLAYERUP @ 1+ 1 AND PLAYERUP ! OTHERFBCTR @
12|FBCOUNTER @ OTHERFBCTR ! FBCOUNTER ! THEN ;
13|DECIMAL -->
14|
15|
+-----Block      104-----
0|( OTHER NEAT SUBROUTINES )
1|: GAME0 GAMEOVER @ EMUSIC E2MUSIC
2|GOFRAME DISPU GOC 5 0 DO 160 82 XY DIPS
3|"A" GAME" COUNT SPOST 120 82 XY DIPS A" OVER" COUNT SPOST EI 30
4|WAIT DI LOOP GAMEOVER ! ;
5|: YOURUP EMUSIC E2MUSIC GOFRAME DISPU
6|GOC 5 0 DO 160 86 XY DIPS A" GET"
7|COUNT SPOST 120 78 XY DIPS A" READY" COUNT SPOST EI 30 WAIT
8|DI LOOP ; DECIMAL -->
9|
10|
11|
12|
13|
14|
15|
+-----Block      105-----
0|( DO A ONE PLAYER GAME )
1|HEX
2|: PLAY1 STARTGAME P2ACT ZERO NPLAYERS ZERO PLAYERUP ZERO
3|GAMEOVER ZERO 3 FBCOUNTER !
4|31 MISSION !
5|BEGIN DOFRAME BUMPMISS GAMEOVER @ END ;
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      106-----
0|( TWO PLAYER GAME )
1|HEX
2|: PLAY2 STARTGAME 1 NPLAYERS ! PLAYERUP ZERO
3|1 FBCOUNTER ! 1 OTHERFBCTR !
4|1 P1ACT ! 1 P2ACT ! GAMEOVER ZERO 31 MISSION !
5|BEGIN
6|OTHERSTATE @ GAMEOVER @ OR IF YOURUP THEN
7|GAMEOVER ZERO DOFRAME 0 FLOOD
8|GAMEOVER @ IF MYSTATE ZERO OTHERSTATE @ IF
9|GAMEO 0 ELSE 1 THEN ELSE 0 THEN
10|SWAPPLAYER BUMPCHK
11|END ;
12|DECIMAL -->
13|
14|
15|
+-----Block      107-----
0|( GAME START UP AND HAVE SOME FUN )
1|HEX
2|: GSU STARTGAME 3141 0 RND# ! 5926 1 RND# !
3|30 MISSION !
4|BEGIN CLRLITES 2 20 OUTP 2 21 OUTP
5|GOFAME 2800 1000 428 A" PUSH 1 OR 2 PLAYER BUTTON"
6|COUNT SPOST
7|4600 4200 428 A" GAME OVER" COUNT SPOST GOC
8|BEGIN RANDOM DROP 10 INP 30 AND 30 <> END
9|10 INP 20 AND 0= IF PLAY2 ELSE PLAY1 THEN
10|EMUSIC E2MUSIC
11|0 END ;
12|DECIMAL ;S
13|
14|
15|
+-----Block      121-----
0|( MASTER CONTROL PROGRAM )
1|HEX ( MISSION START ADDRESSES TABLE )
2|TABLE MSATBL 3300 , 8000 , 9000 , A000 , B000 ,
3|( CRUDE GAME OVER COLORS )
4|DATA MCCOLORS 0 B, 7D B, 0B B, 5A B, 0 B, 7D B, 0B B, 5A B,
5|: GOFAME 0 FLOOD INITMISSIONRAM DRAWMISSIONSCREEN
6|MCCOLORS COLOR DI ;
7|( : DOFRAME INITMISSIONRAM DRAWMISSIONSCREEN BEGIN
8|POLLC IF 1 GAMEOVER THEN JOYSTICK INP SWFIRE AND 0= END DI ;
9|DECIMAL --> )
10|: DOFRAME MISSION 0 31 - MSATBL 0 DUP B0
11|AS = IF 1+ @ DOIT ELSE DROP THEN ;
12|DECIMAL -->
13|
14|
15|

```

```

+-----Block      122-----
0|( DISPLAY PLAYER UP )
1|HEX 428 C= DIPS DECIMAL
2|: DISPU GOFRAME DI
3|240 76 XY DIPS A" PLAYER" COUNT SPOST
4|PLAYERUP @ IF 200 86 XY DIPS A" TWO" COUNT SPOST ELSE
5|200 86 XY DIPS A" ONE" COUNT SPOST THEN ;
6|DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

+-----Block      123-----
0|( OTHER NEAT SUBROUTINES )
1|HEX : BUMPMISS MISSION 1+! MISSION @ 36 = IF SKILLFACTOR 1+!
2|OTHERSKILLF 1+! 31 MISSION ! THEN ;
3|: BUMPCHECK PLAYERUP @ 0= P1ACT @ 0= OR IF BUMPMISS THEN ;
4|: MYSTATE PLAYERUP @ IF P2ACT ELSE P1ACT THEN ;
5|: OTHERSTATE PLAYERUP @ IF P1ACT ELSE P2ACT THEN ;
6|: SWAPPLAYER OTHERSTATE @ IF
7|PLAYERUP @ 1+ 1 AND PLAYERUP ! OTHERFBCTR @
8|FBCOUNTER @ OTHERFBCTR ! FBCOUNTER ! THEN ; DECIMAL
9|: GAMEO GAMEOVER @ GOFRAME DISPU 5 0 DO 160 82 XY DIPS A" GAME"
10|COUNT SPOST 120 82 XY DIPS A" OVER" COUNT SPOST EI 30 WAIT
11|DI LOOP GAMEOVER ! ;
12|: YOURUP GOFRAME DISPU 5 0 DO 160 86 XY DIPS A" GET"
13|COUNT SPOST 120 78 XY DIPS A" READY" COUNT SPOST EI 30 WAIT
14|DI LOOP ; DECIMAL -->
15|

+-----Block      124-----
0|( TWO PLAYER GAME )
1|HEX
2|: PLAY2 STARTGAME 3 OTHERFBCTR ! 1 NPLAYERS ! PLAYERUP ZERO
3|1 P1ACT ! 1 P2ACT ! GAMEOVER ZERO 31 MISSION !
4|BEGIN
5|OTHERSTATE @ GAMEOVER @ OR IF YOURUP THEN
6|GAMEOVER ZERO DOFRAME
7|GAMEOVER @ IF MYSTATE ZERO OTHERSTATE @ IF
8|GAMEO @ ELSE 1 THEN ELSE 0 THEN
9|SWAPPLAYER BUMPCHECK
10|END ;
11|DECIMAL -->
12|
13|
14|
15|

```

```

+-----Block      125-----
0|( GAME START UP AND HAVE SOME FUN )
1|HEX
2|: GSU STARTGAME 3141 0 RND# ! 5926 1 RND# !
3|30 MISSION !
4|BEGIN
5|GOFRAME 2800 1000 428 A" PUSH 1 OR 2 PLAYER BUTTON"
6|COUNT SPOST
7|BEGIN 10 INP 30 AND 30 <> END
8|10 INP 20 AND 0= IF PLAY2 ELSE PLAY1 THEN
9|0 END ;
10|DECIMAL ;S
11|
12|
13|
14|
15|
+-----Block      150-----
0|( CHARACTER VECTORING ROUTINE )
1|SUBR VWRITEC VXL X E LDX, VXH X D LDX,
2|VYL X L LDX, VYH X H LDX,
3|VMAGIC X C LDX, VXPAND X B LDX,
4|VTLL X A LDX, drawchar CALL, RET,
5|
6|SUBR CHARI TBCALC CALL, B PUSH,
7|PQSD E PQS X BITX, 0=, IF, VWRITEC CALL, ELSE,
8|PQSD E PQS X RESX, THEN,
9|B POP, VECTDD CALL, sup CALL,
10|PQSDW PQS X BITX, 0=, IF, VWRITEC CALL, ELSE,
11|PQSD E PQS X SETX, PQSDW PQS X RESX, THEN, KILLOFF JMP,
12|-->
13|
14|
15|
+-----Block      151-----
0|( WONDERFULL TEST CHARACTER INSANITY ROUTINE )
1|HEX
2|DATA BCR 0 80 SETDC -10 0 SETDDC 11 SWAIT
3|120 80 SETDC 11 SWAIT ARET
4|DATA BCR3 3 AREPEAT BCR ACALL ALOOP ARET
5|DATA ACHAR CHARI SETR NULPAT SETP 04 SETXP
6|BF 40 SETS 28 SETM
7|BCR3 ACALL AHALT
8|: TEST STARTGAME GOFRAME MCCOLORS COLOR ;
9|: ZAP TEST
10|3000 1800 ACHAR 43 A2 XYVECTOR DUF WAIT
11|3000 1000 ACHAR 43 A2 XYVECTOR DUF WAIT
12|3000 800 ACHAR 43 A2 XYVECTOR DUF WAIT
13|3000 0 ACHAR 52 A2 XYVECTOR ;
14|DECIMAL ;3
15|

```

```

+-----Block      160-----
0|( FIREBASE EXPLOSION 5 )
1|DATA FBEXP5 6 B, 23 B, QUAD
2|~ 0000 0000 0010 0003 0000 0000 ^
3|~ 3001 3000 1001 1000 0302 0000 ^
4|~ 0020 0030 0200 0020 0000 1000 ^
5|~ 0000 0000 0001 0000 0000 0000 ^
6|~ 3300 2100 2000 3000 0200 0000 ^
7|~ 0030 0200 0002 0000 0033 3000 ^
8|~ 0300 0020 0220 0030 0300 0000 ^
9|~ 0000 0000 0300 0000 0003 0000 ^
10|~ 0033 3002 2200 0110 0011 1000 ^
11|~ 0003 3001 2000 1100 0122 0000 ^
12|~ 0033 2201 2201 1010 1022 1000 ^
13|~ 3300 0221 2011 1100 1110 0000 ^
14|~ 3330 0022 2001 1110 1100 0000 ^
15|~ 0030 2000 0222 2100 0000 1000 ^ -->

```

```

+-----Block      161-----
0|( FBEXP5 CONTINUED )
1|~ 0033 1110 0020 1020 1100 0000 ^
2|~ 0300 0110 0200 1000 0100 0000 ^
3|~ 0001 0003 0030 0010 0030 0000 ^
4|~ 0000 0100 0000 0000 1000 0000 ^
5|~ 0200 0000 0010 0000 0000 0000 ^
6|~ 0001 0030 0000 0030 0010 1000 ^
7|~ 1000 0000 0020 0000 0000 0000 ^
8|~ 0030 0010 0000 0100 0003 0000 ^
9|~ 0300 0010 0000 0000 0000 3000 ^
10|DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      162-----
0|( FIRE BASE EXPLOSION 6 )
1|DATA FBEXP6 6 B, 23 B, QUAD
2|~ 0001 0000 0000 0000 0000 0000 ^
3|~ 0000 0020 0000 0300 0000 2000 ^
4|~ 0000 0000 0300 0000 0010 0000 ^
5|~ 0100 0000 0000 0200 0000 0000 ^
6|~ 0000 0000 0000 0000 0000 0000 ^
7|~ 0200 0010 0031 0001 0020 0000 ^
8|~ 0002 0000 1000 0000 0000 0000 ^
9|~ 0000 0300 0001 0020 0000 3000 ^
10|~ 0100 0000 0000 2000 0000 0000 ^
11|~ 0001 0002 0000 0001 0300 0000 ^
12|~ 3000 0000 0020 0000 0003 0000 ^
13|~ 0020 0200 0000 0000 0010 0000 ^
14|~ 0000 0001 0202 0000 2000 0000 ^
15|~ 2030 1000 0000 0000 0000 0000 ^ -->

```

```

+-----Block      163-----
0| ( FIRE BASE EXPLOSION 6 CONTINUED )
1| ~ 0001 0000 0200 0003 0000 1000 ^
2| ~ 1000 0010 0003 0100 0020 0000 ^
3| ~ 0000 0000 1000 0000 0000 0000 ^
4| ~ 3000 0300 0000 0000 0200 0000 ^
5| ~ 0000 0000 0000 0000 0000 0000 ^
6| ~ 0200 0010 0000 1000 0300 0000 ^
7| ~ 0000 0100 0200 0003 0000 0000 ^
8| ~ 0003 0003 0010 0000 0020 0000 ^
9| ~ 3000 1000 0000 0201 0000 1000 ^
10| DECIMAL -->
11|
12|
13|
14|
15|

```

```

+-----Block      170-----
0| ( NORMAL BCD ADDITION )
1| CODE BCD+! EXX, H POP, D POP,
2| M A MOV, E ADD, DAA, A M MOV,
3| H INX, M A MOV, D ADC, DAA, A M MOV,
4| H INX, M A MOV, 0 ACI, DAA, A M MOV,
5| EXX, NEXT
6| DECIMAL -->
7|
8|
9|
10|
11|
12|
13|
14|
15|

```

```

+-----Block      171-----
0| ( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBCDS )
1| HEX SUBR digit 0F ANI, 0=, IF, D ORA, 0<>, IF, 0F0 A MVI, THEN,
2| ELSE, 0 D MVI, THEN, 30 ADI, EXX, drawchar CALL, EXX, RET,
3| HEX
4|
5| F= DGTL
6| CODE DISPBCDS <ASSEMBLE H POP, M A MOV, H INX, M ORA,
7| H INX, M ORA, A D MOV, 3 E MVI,
8| EXX, B POP, H POP, D POP, X PUSHX, Y PUSHX, EXX,
9| LABEL DGTL Y A MOV, RRC, RRC, RRC, RRC, digit CALL,
10| M A MOV, digit CALL, H DCX, E DCR, DGTL JRNZ,
11| Y POPX, X POPX, NEXT ASSEMBLE>
12| DECIMAL 15
13|
14|
15|

```